THE CHALLENGES OF AUTOMATED METHODS FOR INTEGRATING SYSTEMS

Don Libes, David Flater, Evan Wallace, Micky Steves, Allison Barnard Feeney, and Ed Barkmeyer
National Institute of Standards and Technology
Gaithersburg, Maryland 20899 USA

1. Abstract

Automated methods for integrating systems (AMIS) have been presented as a new approach to solving the dilemmas of multiple and uncoordinated standards, ontologies, legacy systems and the ever-growing cost of traditional integration. This paper explores the challenges of automated methods. By identifying the challenges, we can focus our effort on the areas that are most promising as well as those most likely to fail. We may also contribute to clearing away the hype that distracts, misleads, and ultimately wastes money and labor that is better spent elsewhere.

Keywords: automated integration; ontology integration challenges; legacy integration.

2. Introduction

Integration is traditionally a manual effort. People take existing software and modify or augment it so that it works with other software. In reality, this kind of integration is a very expensive task. By its very nature, manual integration is slow, tedious and error-prone. Manual integration is sometimes non-repeatable as well as hard to document and trace. Such integration can benefit from special purpose tools but they in turn are hard to write and not easily reusable. And once used, they are put aside and suffer "bit rot," meaning that it is unlikely they will work years later should the need arise again.

Manual integration is so expensive that alternatives are frequently chosen. For example, a popular alternative to traditional integration is to throw the existing software away and start over. Starting over has obvious advantages and disadvantages but is outside the scope of this paper [38].

The opposite extreme to manual integration is self-integration. Self-integration would be entirely automated, thereby reducing labor costs. Ideally, self-integration could even achieve integration better than a human by performing computational tasks that are too onerous for any person. Clearly, self-integration is a valuable goal. However, there are several significant unsolved problems that currently prevent self-integration and it is not clear if they are likely to ever become solvable or if they can become solvable at a lower cost than their alternatives.

Although the idea of self-integration as defined in this paper may remain impossibly out of reach, by trying to reach it, we may learn valuable lessons along the way. A significant subgoal is the ability to improve current integration techniques and to achieve partial automation, applying automation wherever appropriate. That is the motivation behind this paper.

Certain companies, standards, or software systems are mentioned in this paper. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technologies nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

3. Key Elements to Automated Methods for Integration

Easy integration is stymied by several major issues.

3.1 Standards

Information technology standards are a key factor in automated integration. We assume the reader is familiar with the merits of standards. For examples, software that complies with a standard gets the benefit of commitments such as application program interfaces (APIs), run-time behavior, etc. Thus, components that adhere to common standards (CORBA [23], SOAP [1], etc.) avoid a large class of problems. For a variety of reasons, however, standards do not solve all problems. Specifically:

- Standards are not perfect; and often have multiple versions.
- Standards often specifically avoid addressing certain issues.
- There is sometimes a choice of conflicting standards.
- There is always a new standard waiting in the wings.
- Technology is faster than any standards-setting body.
- Standards sometimes inhibit innovation.
- Companies sometimes see standards as preventing a competitive advantage.

Standards are a mixed blessing. For rapidly advancing technology, standards can only reflect a snapshot, often one that is blurry in areas. Indeed, standards come about because there are differences that cannot be resolved in any other way. So people work together to create a standard; the standard, if timely, serves its purpose for a short time; then as the standard fails to address new extensions, work starts all over on a new standard.

An example of the relentless and frequent superseding in standards can be seen in the progression of SGML (Standard Generalized Markup Language) to HTML (HyperText Markup Language) to XML (Extensible Markup Language). In a relatively short time span, each has built on the previous, representing a recognition of inadequacy and/or incompleteness in prior standards. Along the way, there have been numerous branches resulting in detours and deadends, many that became real standards and even more that did not. Both types becoming obsolete, in most cases before the paper they were printed on was dry.

There are many other reasons that standards have not solved the integration problem and for some cases, standards simply cause more problems. Suffice to say that standards are no silver bullet. Indeed, inadequate deployment of truly useful standards is one of the driving forces for automated integration.

3.2 Ontologies

Ontologies commit to a particular conceptual model and in a sense, effectively behave as a standard. Because ontologies are effectively one type of standard, ontologies share some of the same problems as standards noted above. Ontologies have additional problems:

- Conflicting ontologies, well, conflict. Conceptualizations can be incompatible.
- Ontological coverage is incomplete. There is no universal ontology; rather there are islands of ontologies.
- New ontologies are being developed incessantly. In existing ontologies, there is "conceptual drift" over time.
- Few true ontology experts exist and even such experts disagree over the same ontology.
- There is a lack of maturity of ontologies and ontological tools.
- Different ontologic systems capture different interests and have different inferencing biases: causality, time, actions, negation, etc.
- Translations between ontologies can be very hard, potentially requiring expertise in two entirely different ontological systems.

This last bullet weighs significantly on the future of integration automation. Specifically, what is the justification for the belief that ontological translation can be automated? There is a handful of such research projects [39] [26] and positive results are limited to very small or toy examples.

For example, consider the concept of color. Color can be represented as red, green, and blue components. Or cyan, magenta, and yellow. Or hue, saturation, and brightness. An automated translator might assume "color" equals "color" in two different ontological systems. The words match, the semantics match (to a degree), and even the number of components is the same. Of course, the semantics of the components are not precisely the same and any attempt to automatically match them would be wrong for some (but not all) applications.

Indeed, maps between color models are not necessarily complete and the field of "gamut mismatch" addresses this area but choosing the algorithm or solution still requires manual consideration of the context [12]. The problem is that even the semantics of color as a whole are not equivalent from one model to another. For example, what a human considers as color is just a subset of what some animals see or hardware sensors see.

Contrariwise, a semantic analysis of two concepts that truly are in agreement can (and generally will) still fail because some aspect of the representations are different. For example, time periods are represented as *from/to* in CIDX (Chemical Industry Data Exchange) while in the same OAGIS (Open Applications Group Integration Specification) concept is represented as *from/duration*. It is unlikely that such conceptual mismatches could be automatically discovered and mapped [8][25].

By comparison, forced single standards may have a better financial return. For example, during the 2002 merger of Comcast and AT&T Broadband, the newly merged company faced integration of multiple billing systems but opted for a one-time conversion despite the significant financial expense due to contractual obligations [15].

3.3 Legacy Systems

We are interested in integrating both modern systems as well as legacy systems. It is impossible to rigorously define the difference between modern and legacy systems but some generalizations are possible. Modern systems have models, use modern standards, and have working development environments. Although integration of modern systems present certain challenges, integration of legacy systems provide additional difficulties:

- Semantics and models are typically lacking and unrecoverable from legacy systems.
- Source code or original development environments may be unavailable for integration.

- Legacy systems by their very nature are poorly supported if at all.
- Modern systems may be altered. In contrast, legacy systems must generally be wrapped.

By their very nature, legacy systems have stepped out of the mainstream of development. Development personnel may have long since disappeared with the knowledge that was only in their mental models. Integration projects involving legacy systems may require reverse engineering efforts at large expense.

In efforts to construct models for legacy systems, it is not surprising to find that they are self-contradictory, in part because they were built without models and thus, in essence, incorporate a variety of conflicting models with conflicting conceptualizations. Some of these conceptualizations may be due to conflicting system requirements, often due to incompatibilities introduced over the life of a system.

For example, the Expect package [17] originally had its own I/O subsystem because the underlying I/O subsystem did not support the null character, a restriction in the C library itself. It was thus impossible to carry out certain operations that required the services of both subsystems, such as having Expect pattern-match across a regular file [18]. This was fixed at great expense much later by backporting Expect's I/O subsystem but could have been avoided in the first place by having a single model to expose and unify the multiple sets of restrictions.

4. State of the Art of Automated Methods

Automated methods have been proposed as possible solutions to solving the dilemmas of multiple and uncoordinated standards, ontologies, legacy systems, and the ever-growing cost of traditional manual integration. Automated methods would save any effort that has already been expended while allowing the use of newer software and standards. Automated methods would reduce labor costs. Ideally, automated methods could even achieve integration better than a human by performing computational tasks that are too onerous for any person.

Automated methods are intended to reduce the traditional costs of manual integration. If sufficiently automated – essentially self-automated – they would allow systems to seek out new opportunities on their own.

4.1 Universal Adapters

Various projects have proposed the idea of universal adapters [14] [21]. At the present time, such adapters are libraries of interfaces in integration development testbeds and thus function as partially pre-integrated middleware. Such universal adapters are far from fully automatic but generally have a clear delineation over what protocols and APIs they do support. For legacy systems that use no standard interfaces, universal adapters fall back to manual integration.

4.2 Model-Driven Architectures

Model-driven architecture (MDA) such as OMG's MDA promises technology-independent development [32], thereby avoiding idiosyncrasies of implementation details and instead allowing focus purely on the high-level concerns. In theory, MDA tools can generate SOAP implementations, CORBA implementations, etc., whatever is appropriate, all from the same models. This holds promise for integration of future systems but it is unclear what leverage this can bring to bear against legacy systems that have no such unified or platform-independent model. And legacy systems that do have models generally only had those models referenced during design and not during implementation so that there are mismatches to the true model.

4.3 Universal Ontologies

Universal ontologies presume a framework of all ontological knowledge or at least a scaffolding against which new knowledge and other ontologies can be placed in perspective. Although there are several ongoing projects (SUMO (Suggested Upper Merged Ontology), OpenCyc, etc.) that offer themselves as such scaffolding, the effort in their construction is vast and it is unclear whether it will be possible to automate maps between them or, for that matter, to less complete ontologies. In reality, most software systems are not based against any formally-constructed ontology and will need human labor to construct manual maps. Ontology mapping is an area that is far from mature.

Many people question whether it makes sense to have a truly global ontology in the sense that its own complexity will prevent it from being useful because there will be too many relationships that must be dealt with unnecessarily. A second issue is that there are valid reasons for having different structures of knowledge. As an example, the nervous system of a mosquito contains approximately 150 neurons with a limited amount of memory. Yet, simulating the flight of a mosquito is a daunting task for any modern computer despite having data storage and processing elements that are thousands of times more powerful than those of a mosquito. Clearly, the processing structure of a mosquito and indeed, most biological systems, is radically different than what is produced by traditional knowledge engineers for ontologic purposes such as deduction and reasoning.

4.4 Integration Tools and Research

Below is an assortment of tools and research projects specifically aimed at integration automation. It is not meant to be complete – indeed, it is a small fraction of the universe of works – but merely to whet the appetite of the reader and to fill in the discussion with some real examples. Further discussion on related tools, techniques, and comparisons of them can be found elsewhere [24][28][35][36].

4.4.1 AMIS

The AMIS project [5] at the National Institute of Standards and Technology is studying such automated methods for integration. AMIS hopes to derive implicit ontologies from legacy systems which can then be used in the creation of expert systems that can generate process-specific wrappers. These wrappers would then transform the physical message sequences on the basis of equivalent business notions.

4.4.2 Contextia

Modulant's Contextia [20] provides a semantic mediation framework that is middleware for the interoperability of information among disparate systems through explicit and formal representation of incompatible, conflicting semantics. The Contextia Interoperability Workbench maps semantics to an Abstract Conceptual Model (ACM) to perform transformations at runtime. Contextia focuses on PDM (Product Data Management), ERP (Enterprise Resource Planning), SCM (Supply Chain Management), CRM (Customer Relationship Management), and related applications.

4.4.3 Contivo

Contivo [9] automates the design of data transformation between applications through reuse and collaboration. Contivo's Analyst builds and applies transformation commands between business objects. It is partly human-interactive, using visual maps, and partly self-learning. The focus of Contivo is interface models and commercial frameworks such as BEA, Tibco, and J2EE (Java 2 Platform, Enterprise Edition) [1][37][34].

4.4.4 GKB

The GKB-Editor [20] (Generic Knowledge Base Editor) from SRI International, is a tool for editing ontologies and knowlege bases using a graphical user interface across varied frame-representation systems. The GKB interface represents objects and data items as a graph, with the relationships as edges.

4.4.5 MIST

The Carnot [39] project tackled the problem of logically unifying physically distributed, enterprise-wide, heterogeneous information. Part of Carnot was the Model Integration Software Tool (MIST), which provides a graphical user-interface to aid a user in the integration of different databases via a unified enterprise ontology.

4.4.6 InfoSleuth

The InfoSleuth project [4], based on MCC's Carnot technology, provides tools to find information in related and unrelated networks. InfoSleuth uses agents to represent users, ontologies, and information sources such as query engines. InfoSleuth uses interaction templates for mapping interactions.

4.4.7 Ontobroker

Ontobroker [26] is an outgrowth of work done at the University of Kalrsruhe. Ontobroker is an inferencing engine and query system. The engine manipulates statements using a subset of first-order logic. It is middleware intended to integrate heterogeneous data sources (with focus on e-commerce). Available is a limited number of drivers to standard exchange formats (e.g., RDF (Resource Description Framework)) and data sources (e.g., MS SQL (Microsoft Structured Query Language), Sybase). A related tool, OntoEdit, is a graphical "ontology neutral" editing tool which does translation to/from DAML+OIL [7], and RDFS [16] and call also read from several SQL-based databases (e.g., MS SQL, Sybase).

4.4.8 Ontolingua

Ontolingua [13] is a set of tools for translating ontologies, written at Stanford University. KIF [19] (Knowledge Interchange Format) provides an exchange language, portable over a variety of representation systems. Translators from KIF into various representation systems are available.

4.4.9 OntoMap

The Ontomap [27] project is an approach for ontology translation that use human-guidance to aid the semi-automated process. Ontomap starts with an informal process and then becomes successively more refined through a series of formal verification steps such as consulting the WordNet database. Ontomap is implemented using Jess, the Java Expert System Shell and uses XML and NIST's PSL [22] as a ontology interchange format.

4.4.10 OntoMorph

Ontomorph [6] is a project/environment aimed at solving the ontology translation problem. It performs syntactic

translation based on pattern matching of syntax trees and ontologic-based semantic rewriting. Ontomorph is based on PowerLoom for knowledge representation and PLisp ("Pattern Lisp") for concise specification and destructuring of translations.

4.4.11 TSIMMIS

The TSIMMIS Project [27] was created to develop tools that integrate heterogeneous information sources. TSIMMIS modules obtain properties from unstructured objects and are able to map information and constraints across heterogenous sites into a uniform data model.

5. What Needs To / Might Be Done in the Future

More global ontologies and maps between ontologies are certain to be constructed in the future. We can also look forward to more complete logical relationships so that reasoning can be done both inter-model and intra-model. Rather than filling holes in models, the integration process will then be reduced to making maps between ontologies. Rahm and Bernstein provide a good survey to schema matching [29].

Map making may be partially automated with the creation of smarter modeling tools. Such tools could derive semantic meanings from context or other implementation information that is currently ignored. It may be possible to automate parts of model recreation from legacy software. This would substantially reduce the cost of legacy software integration.

If MDA comes to fruition, this would greatly simplify later integration projects as such projects would essentially throw away brittle implementations and start integration work directly from the models.

Further work needs to occur in the area of interaction ontologies [10]. Traditionally, maps have related static concepts between ontologies. The maps between the static concepts and the APIs are as significant and necessary as the static maps.

6. What Will Not Be Done ... For a Very Long Time

For a variety of reasons, it is difficult to envision total automation of software integration in the general sense.

It is human nature that system creation, whether legacy or anew, inevitably incorporates unmodelled constraints and that these constraints will be inaccessible by any automated reasoning engine. In such systems, the issue is not how to automate the ontology mapping but how to discover that the ontologies themselves are incomplete or incorrect with respect to the implementation. An automated solution to this problem would require the equivalent of a human who occasionally makes mistakes, guesses, asks questions, and expends painfully costly resources exploring dead ends.

In a formal sense, the act of integration imposes new, possibly un-modelled constraints that may not be met by the existing systems [11]. For any given abstraction, it is possible to construct an integration scenario in which a failure will occur because of some property that was not explicitly modeled. For example, a reasonable abstraction of time-of-day constrains seconds to a range of 0 to 59. A system with such a model may integrate properly with many other systems or, at least, give the appearance of successful integration – even for many years. However, a formal representation of seconds in the Coordinated Universal Time Scale allows for the insertion of additional seconds to account for the physical changes in how time is synchronized to movement of the solar system. Representation of such 'leap seconds' requires an expanded range. So an integration may apparently succeed only to fail come a New Years many years distant from the integration itself.

The need for the abstractions to take an explicit stance with respect to otherwise-irrelevant properties only arises when integration is attempted. Yet by virtue of numerous undocumented and/or un-thought-of implementation details, any realizations of these abstractions in engineered artifacts such as software implicitly take stances with respect to all properties. It should not be surprising that when confronted with such properties, integrations fail.

Even in the best modelled of worlds, the continuing growth and change of ontologies will remain a problem. Indeed, as ontologic systems become better at incorporating relationship information, it is likely that the different ontological 'standards' will become ever more 'different' as they incorporate subtle new meanings. This type of change happens frequently and we live in an age when the rate of change is ever increasing. This fear of the inability to handle change in ontologies is captured by Sowa [33]:

> *"I expect the evolution of ontologies to recapitulate our experiences with subroutine libraries. There will inevitably be libraries of them with version numbers that are periodically updated. Any attempt to avoid having multiple versions of the "same" ontology on the same system at the same time will undoubtedly create a new Hell, not unlike DLL Hell."*

7. Conclusion

Automated methods for integrating systems have been presented as a new approach to solving the dilemmas of multiple and uncoordinated standards, ontologies, legacy

systems and the ever-growing cost of traditional integration. Existing approaches for automated methods are interesting but are strongly suggestive that automated integration is not possible for large-scale software. For the foreseeable future, it is likely that there will always be a human component and that humans and semi-automated translators will work together. Methods for semi-automated integration are more promising although they have their own disadvantages. It is worth continuing research and experimentation with automated and semi-automated integration to solve pieces of the general problem of automated integration.

8. References

[1] BEA Systems, http://www.bea.com, 2003.

[2] Box, Don, et al. *Simple Object Access Protocol (SOAP) 1.1*, http://www.w3.org/TR/SOAP/, W3C Note 08 May 2000.

[3] Chaudhri, Vinay and Lowrance, John, *Generic Knowledge-Base Editor*, http://www.ai.sri.com/~gkb

[4] *The InfoSleuth Agent System*, http://www.argreenhouse.com/InfoSleuth

[5] Barkmeyer, Edward , Barnard Feeney, Allison , Denno, Peter , Flater, David , Libes, Don , Steves, Michelle Potts, and Wallace, Evan, *Concepts for Automating Systems Integration*, NIST IR 6928, (2003).

[6] Chalupsky, Hans, *OntoMorph: A Translation System for Symbolic Knowledge*, USC Information Sciences Institute, http://www.isi.edu/~hans/ontomorph/presentation/siframes.html

[7] Connolly, Dan, et al., *DAML+OIL (March 2001 Reference Description*, http://www.w3.org/TR/daml+oil-reference, W3C Note 18 December 2001.

[8] Chemical Industry Data Exchange, CIDX, http://www.cidx.org, 2002.

[9] *Contivo – Enterprise Integration Modeling*, http://www.contivo.com, 2001.

[10] Denno, Peter, Steves, Michelle, Libes, Don, and Barkmeyer, Ed, Model Driven Integration Using Existing Models, to appear in *IEEE Software*, 2003.

[11] Flater, David, A Logical Model of Conceptual Integrity in Data Integration, submitted to *The NIST Journal of Research*.

[12] Gentile, R. S., Allebach, J. P. and Walowit, E., A comparison of techniques for color gamut mismatch compensation, *Proc. SPIE Human Vision, Visual Processing, and Digital Display*, B. E. Rogowitz, Ed., 1989, vol. 1077, pp. 342-354.

[13] Gruber, T. R., A Translation Approach to Portable Ontology Specifications, *Knowledge Aquisition, 5(2)*, 199-220, http://www.ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html, 1993.

[14] Hendrick, Stephen, and Hendrick, Kathleen, *iWay Software: A Pragmatic Solution to Application Integration Needs*, http://www.iwaysoftware.com/pdf/idc_analyst.com, November 2002.

[15] Joyce, Erin, CSG, *Comcast Billing Snafu Grows*, http://boston.internet.com/news/article.php/1558351.

[16] Lassilla, Ora, and Swick, Ralph, eds., *Resource Description Framework (RDF(S)) Model and Syntax Specification*, W3c Recomendation, 22 February 1999.

[17] Libes, Don, *Exploring Expect*, O'Reilly, Sebastopol, CA, January 1995.

[18] Libes, Don, Writing a Tcl Extension In Only ... 7 Years, *Proceedings of the Fifth Annual Tcl/Tk Workshop '97*, Boston, MA, July, 1997.

[19] http://logic.stanford.edu/kif/dpans.html

[20] http://www.modulant.com

[21] NexPrise, Inc., http://www.nexprise.com, 2003.

[22] NIST, *Process Specification Language (PSL)*, http://ats.nist.gov/psl, 2002.

[23] Object Management Group, *OMG's CORBA (Common Object REquest Broker Architecture) Website*, http://www.corba.org, 1997-2003.

[24] Noy, Natalya, and Musen, Mark, Evaluating Ontology Mapping Tools: Requirements and Experience, SMI-2002-0936, http://www.smi.stanford.edu/pubs/SMI_Reports/SMI-2002-0936.pdf, *Workshop on Evaluation of Ontology Tools at EKAW'02 (EON2002)*, 2002.

[25] Open Applications Group, http://www.openapplications.org, 2003.

[26] http://ontobroker.aifb.uni-karlsruhe.de/index_ob.html

[27] http://www.ontomap.org

[28] Perez, Asuncion Gomez, ed., A survey on ontology tools, http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-3.pdf, *Ontology-based information exchange for knowledge management and electronic commerce, IST-2000-29243*, Deliverable 1.3, 2002.

[29] Rahm, Erhard and Bernstein, Philip A., *A Survey of Approaches to Automatic Schema Matching, The VLDB Journal 10:334-350 (2001)*, ftp://ftp.research.microsoft.com/pub/tr/tr-2001-17.pdf, 2001.

[30] http://www.smi.stanford.edu/pubs/SMI_Reports/SMI-2002-0936.pdf

[31] Rys, Michael, *TSIMMIS (The Stanford IBM Manager of Multiple Information Sources)*, http://www-db.stanford.edu/tsimmis, April 4, 1998.

[32] Siegel, Jon, Making the Case: OMG's Model Driven Architecture, *SD Times*, http://www.sdtimes.com/news/064/special1.htm, October 15, 2002.

[33] Sowa, John, F., *Re: SUO: Enlightened Semantic Web, http://suo.ieee.org/email/msg09078.html*, March 20, 2003.

[34] Sun Microsystems Inc, *Java 2 Platform, Enterprise Edition (J2EE)*, http://java.sun.com/j2ee, 2003.

[35] Sure, Y., Corcho, O. and Angele, J. (eds.). Evaluation of Ontology based Tools (EON2003), *Proceedings of the 2nd International Workshop EON2003, Workshop at the 2nd International Semantic Web Conference (ISWC 2003)*, 20-23 October 2003, Sundial Resort, Sanibel Island, Florida, USA.

[36] Sure, Y. and Iosif, V. First Results of a Semantic Web Technologies Evaluation, *Proceedings of the Common Industry Program at the federated event co-locating the three international conferences: DOA'02: Distributed Objects and Applications; ODBASE'02: Ontologies, Databases and Applied Semantics; CoopIS'02: Cooperative Information Systems DOA/ODBASE/CoopIS'02*, October 28 - November 1, 2002, University of California, Irvine, USA, pages 69-78.

[37] Tibco Software Inc., http://www.tibco.com, 2003.

[38] Weinberg, Gerald, *The Psychology of Computer Programming*, Van Nostrand Reinhold Company, 1971.

[39] Woelk, D., Cannata, P., Huhns, M., Shen, W., and Tomlinson, C., Using Carnot for Enterprise Information Integration, *Second International Conference on Parallel and Distributed Information Systems*, http://www.mcc.com/projects/carnot, January, 1993.