Systems Engineering Lessons from OpenII

Len Seligman, Arnon Rosenthal, M. David Allen, Maya Li, Catherine Macheret, Peter Mork, Ken Smith, Chris Wolf

Last update: March 25, 2010

Open Information Interoperability (OpenII) is a collaborative effort spearheaded by MITRE and Google to produce an extensible, open source information interoperability toolkit [Seligman]. OpenII makes it faster and cheaper to build information exchanges when the sending and receiving systems differ in how they represent related information. Because the software is all open source, it can be freely downloaded and customized to meet specific organizational needs. OpenII is being used to solve diverse government information sharing problems. Also, several additional homeland security and defense organizations are actively exploring its use.

The purpose of this short paper is to describe the systems engineering lessons learned from the experience of building and applying OpenII. These lessons fall into in three primary bins:

- Design of information interoperability tools and tool frameworks
- Lessons for engineers working on interoperability problems
- Lessons for enterprise planners

Design of Information Interoperability Tools and Tool Frameworks

Use an open, plug-in architecture. Prior information interoperability (II) tools have these limitations: (1) they only address a narrow part of the II problem (e.g., identifying candidate matches without helping generate data exchange code), or (2) they use a proprietary infrastructure that does not interoperate well with other vendors' tools. In contrast, OpenII has a common infrastructure with open, well-specified interfaces that allow multiple tools to share knowledge. As a result, it is easy for different organizations to plug in their own tools. In addition to supporting interoperation among separately designed II tools, the individual tools have been designed to be easily customizable. For example, the Harmony schema matcher has several built-in alternate match algorithms and new ones can be inserted easily. In addition, OpenII provides a choice of multiple back-end databases (Postgres and IBM's Derby) for the metadata respository and could be quickly extended to support others, such as Oracle or SQL Server.

In order to avoid technology-specific stovepipes, capture as much knowledge as possible in a model-neutral way. Many II tools are specific to a particular technology. For example, Altova's MapForce creates mappings from one XML source to another and produces XSLT data exchange code. Unfortunately, it does not capture the mapping knowledge in a way that is usable by efforts that use different technologies, such as SQL-based data exchange among relational databases. Without a technology-neutral repository for schemas and mappings, many opportunities will be lost to reuse and repurpose previously specified mappings. This is why OpenII has importers and exporters for various technologies (including XML Schema, relational

databases, OWL ontologies, and Excel spreadsheets), but it maintains all schema and mapping information in a common framework. Similarly, OpenII tools do as much work as possible in a neutral metamodel. The Harmony schema matcher, for example, operates over schemas in the neutral model, which supports matching of any model for which OpenII has an importer.

Provide a common framework with tools for the entire II lifecycle. Most existing tools begin with the source and target data formats already identified and do not help with data source discovery and broad information space awareness (i.e., what are the natural clusterings of data sources which may be candidate sharing partners?). In addition, most II tools support only one phase of the II process, without an easy way to pass the knowledge to downstream tools. In contrast, OpenII begins earlier in the life cycle by providing an innovative discovery capability over structured data sources (i.e., Schemr [Chen]) and a tool for quickly identifying the major clusterings of an enterprise or community's structured data assets (i.e., Affinity [Smith]).[1] In addition, by sharing all knowledge through a common repository, OpenII provides a natural mechanism for passing knowledge to downstream tools. For example, once interesting clusters are identified and overlaps explored through Affinity, one can drill down to explore detailed correspondences using Harmony, add functions and ultimately generate data exchange code using RMap or XMap.

To ease tool adoption, require minimal configuration. The initial version of OpenII provided only one way to host the schema and mapping repository: on a Postgres relational database management system (DBMS). While a multi-user DBMS has big advantages for enterprise use (e.g., to enable sharing and reuse of mappings), there is also a downside—namely, that it requires expertise to install and maintain the database. During the initial kickoff meeting for OpenII, however, we learned of IBM's Derby, an open-source, very lightweight, single user DBMS. We decided to create an alternate version of the tool suite which has Derby preconfigured in it. While some of the enterprise level sharing features are then not available, this enables integrators and in some cases even business analysts to just unzip a file and begin integrating systems with no configuration necessary. This has greatly lowered the barrier to using OpenII. Another lesson is to not only provide a low barrier to entry but also to provide an easy migration path to enterprise level capabilities. If for example an interoperability project has been kicked off with the lightweight, Derby-based version of OpenII, it is fairly straightforward to migrate the captured knowledge to the enterprise version.

Make easy things easier and hard things possible. We have used this as a general design guideline, and we believe it has served us well. The idea is to define a clean interface for simple tasks, using defaults or a "More Options" capability to hide complications needed for advanced uses. For example, the Harmony schema matcher offers several different algorithms for suggesting candidate correspondences across data sources, including bag-of-words matching, edit distance, exact structure matching, bag-of-words with a thesaurus, etc. Depending upon the details of the specific match problem, different ones of these algorithms will perform better. The good news is that this allows the tools to be customized to work well for a particular environment. The bad news is that most integrators will have no idea which algorithm to choose. The solution is to allow the customization but to provide a default configuration that works

[1] Affinity was built by the Common Ground MSR and contributed to the OpenII tool suite. Schemr was built jointly by Google, Inc., and the University of California at Berkeley.

reasonably well on most problems. In addition, organizations can customize the tool with their own defaults that better match their environment.

Design your tools to give integrators visibility into the state of the interoperability effort, including progress as well as highlighting potential bottlenecks. In addition, early feedback on the difficulty of an II task would be tremendously valuable to integrators as well as enterprise planners.

Lessons for engineers working on interoperability problems

Good tools save time and money. While there is often a tendency to "just hack it up" with skilled programmer sweat, those in charge of an information interoperability effort should first determine what tools might speed the effort. We have seen the beneficial effect of tools on several II efforts. For example, one government program office needed to determine which items across several large code lists corresponded with one another. The previous state of their practice had been to map these items manually in an Excel spreadsheet. By using OpenII's Harmony matcher, most of the mappings were determined automatically in minutes or seconds. Analysts were freed to focus on the tough cases that actually required human expertise; the time required to complete the mapping was reduced substantially.

In addition, the right tools can empower subject matter experts without technical expertise to contribute their knowledge. Finally, tools can have another, non-technical benefit. Often, the biggest barriers to information sharing are political and cultural. By showing a decision maker that tools are available, this can help remove excuses and reduce resistance to sharing.

Make data standards modular, so that users don't need to understand a large, complex artifact if they only need a small piece. A big reason for the success of Cursor on Target was its extreme simplicity. In contrast, some substantially more complicated data standards have achieved only limited adoption because of their complexity. This argues not only for simple "cores" but also for *layered complexity*, where you can choose the granularity appropriate to the characteristics of a community.

Describe intersystem mappings and auto-generate data exchange code from those mappings instead of just coding. There are a number of reasons for this. First, given good tools, this saves work and also reduces the skill level required to create data exchange code. Second, the relationships are better documented than in just code. This documentation is extremely valuable whenever II code must be changed or if it is to be repurposed to a new environment.

Seek to develop a repeatable process and not to just produce artifacts. It is important to have a CONOPS for employing and extending data standards. In addition, education in the proper use of standards will increase their value. Finally, seek scalable processes. One popular process—"we got everyone in the room and locked the door until a standard emerged"—does not scale. To scale to larger groups, provide web-accessible, enterprise scale repositories and discovery tools, as well as II tools that exploit those repositories. (Metadata repositories without II tools that exploit the information in those repositories are of limited value.)

Lessons for enterprise planners

Recognize that heterogeneity is a permanent fact of life; data standardization, while very useful within communities of interest, cannot eliminate diversity. No monolithic data standard will describe all systems in a large enterprise with many autonomous participants, continuous change, mergers and splits, and 20 years of coexisting technology. Instead, multiple standards are inevitable. While we should minimize diversity where possible, we also need to help system builders deal with it.

Interoperability planning should include both internal and external partners. These external partners (industrial, other government agencies, coalition partners) may use different standards.

Set realistic interoperability goals that really help your planning. Too many briefings contain unrealistic goals like "Seamless data sharing with anyone", with the implication that the end state is reachable. More appropriate goals are: "Robust data sharing with important partners" plus "Have the tools and capture the knowledge so you can quickly create new sharing arrangements."

Be a servant, not a policeman; realistically consider the incentives of key participants. For example, if a key part of an enterprise strategy is adoption of a data standard, consider if individual programs will see value in adopting the standard. If not, they will often find ways to comply with the letter but not the spirit of the mandate, with the result that the intended interoperability benefits are not achieved.

Provide free tools. Wherever there is a preferred data standard, provide free tools that both 1) ease adoption (including mapping to the standard) and 2) provide real interoperability benefits (e.g., enable you to more easily map to others, whether conformant or not). OpenII supports this strategy nicely, since it is open-source, freely downloadable, and can be customized for particular community requirements.

Do pilots and real evaluation of costs and benefits before rolling out sweeping strategies and mandates. There is often a gigantic gap between a high-level vision on a briefing chart and the implementation of the vision. Real evaluation of tools and strategies is critical and should include a success measure. Many pilots fail to adequately measure whether the approach being piloted provided a real savings, or did it just demonstrate that programmer sweat can achieve a lot.

Open your interoperability framework to all: industry, academics, etc., not just registered users or US citizens. While instance data is often very sensitive, schemas and mappings seldom are. Opening the framework and repositories to as many participants as possible brings orders of magnitude more talent to bear on government problems than if we needlessly restrict access.

Summary

We have described the systems engineering lessons learned from the experience of building OpenII and applying it to several government interoperability efforts.

This is a living document. We plan to amend it as we learn additional lessons. Please contact the authors at seligman@mitre.org with any comments or suggestions.

References

[Chen] K. Chen, J. Madhavan, and A. Halevy, "Exploring Schema Repositories with Schemr," *International Conference on the Management of Data (SIGMOD '09)*, May 2009

[Robbins] D. Robbins, "Unmanned Aircraft Operational Integration Using MITRE's Cursor on Target," The Edge, Vol. 10, No. 2, http://www.mitre.org/news/the_edge/summer_07/robbins.html, Summer 2007

[Seligman] L. Seligman, P. Mork, A. Halevy, K. Smith, M. Carey, K. Chen, D. Burdick, C. Wolf, J. Madhavan, A. Kannan, "OpenII: An Open Source Information Integration Toolkit," *International Conference on the Management of Data (SIGMOD '10)*, May 2010

[Smith] K. Smith, C. Bonaceto, C. Wolf, B. Yost, M. Morse, P. Mork, D. Burdick, "Exploring Schema Similarity At Multiple Resolutions," *International Conference on the Management of Data (SIGMOD '10)*, May 2010

