Arriving at a Service-Oriented Architecture As a Solution for Integrated Justice Information

Table of Contents

The author of this report welcomes comments, questions, corrections, or any other feedback.

Fred Lengerich
System Architect
Justice and Public Safety
Science Application International Corporation (SAIC)

Telephone: (303) 713-1582
E-mail: lengerichf@saic.com

1. Executive Summary

Gartner Research estimates that 40% of a software projects effort is spent on interfacing the software with other software systems. The costs of duplicate data errors and re-keying of data has been well discussed in many papers on integrated justice systems from SEARCH, IJIS Institute, BJA, and others. The mechanisms of software integration are therefore critical to solving the justice information integration problem. This paper builds the background necessary to understand the business advantages and disadvantages of any integration architecture.

This paper takes any level of reader through a primer of what is information integration and then moves on to present the various data architectures for integration. Against this background the system architecture of a Service-Oriented Architecture (SOA) is detailed.

The paper proposes that the system architecture of SOA provides the best model for the vagaries found in justice integration projects. SOA is that next major evolution in architecture that leverages the work of web services and XML, which are becoming common in justice integration today. Both web services and XML are capabilities, not stopping points, which help to enable the larger requirements of an integration architecture that is cost-effective, flexible, and future-proof. For justice integration, that architecture is SOA. SOA not only natively supports integration capabilities, it also allows for rapid process and workflow analysis and modeling which are critical in these times of watchful budget control and need for real-time access to complete information. The paper shows how the full value of the work of SEARCH and the XML model GJXDM are realized only in a service-oriented architecture.

2. Introduction

We need better information integration between our justice systems today. This headline statement is old news to the justice community. What is news is how fast integration technology has matured since being introduced in the early 1990's. Integration has been one of the top four investment areas of CIO's since 2000 as surveyed by Morgan Stanley each quarter. Just because the integration of information systems is a major priority does not make it a simple task. Integration solutions are much harder to implement than applications[1] because integration involves the total technology environment. The concepts for integration are actually very simple; it is the magnitude that makes the task a challenge.

[1] Application(s) – another term for a computer system or program, an example of an application would be a court scheduling program, criminal history system, or jail management system.

3. The Fundamentals of Integration

3.1 *Integration patterns*

A review of Justice Integration projects and planned projects over the last 5 years reveals that the scope of what information integration means in the justice community is very broad. The integration patterns encountered can be organized into the following categories:

1. Virtual query or portal
2. Data warehouse
3. Centralized index (lookup table)
4. Justice application suite
5. Enterprise Application Integration (EAI)

A starting point for justice integration is to define application integration. The author's research has shown that "data politics" defines which integration pattern is used in nearly every situation, rather than the capabilities or needs of the operation or technology. Data politics is defined as the willingness and political ability of one organization to share data with another organization.

Integration patterns vary by the level of information integration. Table 1 describes the range of information integration in contrast to the integration pattern.

Integration Pattern	Information Integration Capability	Application data is updated automatically (based on business rules)	Only one copy of the data	Always accessing the most current information	Defines a definitive Source for a data element	Enables process or workflow improvements
Virtual Query/Portal	Low	No	Yes	Yes	No	No
Data warehouse	Low	No	No	No	No	No
Centralized Index (lookup table)	Low	No	No	No	No	No
Justice Application Suite	High	Yes	Yes	Yes	Yes	No
Enterprise Application Integration (EAI)	High	Yes	Yes	Yes	Yes	Yes

Table 1

3.1.1 Description of common Justice Integration Patterns

Virtual Query (or Portal)
The typical implementation of a virtual query is to access a small set of data from a select few justice applications and then making the results available in an internet browser. Examples of the typical type of data queries are: retrieving a mug shot, SID lookup, or inmate locator.

Advantages:

- Virtual queries give a quick visible fix to the need to provide access to justice information.

Integration Limitations:

- If what you want is not in the query, you need to make a request to have it programmed.
- There is no system-to-system integration, so inconsistent information between systems is still possible.
- Does nothing to solve the re-keying problem of justice data.
- Single threaded to a web server (which will display the query pages), which if development is sloppy will become a rat's nest of transformation logic needed for each accessed application.
- No ability to examine and improve business process or workflow

- Most virtual queries projects start out with modest needs and then the needs grow. This causes the source applications to be revisited many times to "tweak" the interface to get just that next piece of data. Each new tweak takes longer since there is no structured plan to add tweaks.
- Software must be built or acquired to transform the many data sources into a useable format.
- There is no event notification when data changes; the requestor must constantly re-query to get the latest information.

Data Warehouse

A data warehouse can be considered to be the old form of a virtual query. In a virtual query, data can be assimilated from many applications on the fly and never stored. A data warehouse assimilates information from many sources but stores the information in a database for later retrieval.

Advantages:

- The native applications are in tight control of their data.

Integration Limitations

- A data warehouse makes two copies of the information; there is now a data currency issue.
- Building a data warehouse is not a quick process since a database must be developed.
- The data in the warehouse is only as "fresh" as the last data warehouse load. When you access the data warehouse, how do you know that the data you have is the most current? What happens if the last data warehouse load failed?
- If what you want is not in the data warehouse, you have a major development effort to get it in the query; data extraction code, transformation code, redesign the data warehouse, and update data warehouse query tool.
- There is no system-to-system integration, so inconsistent information between systems is still possible.
- Does nothing to solve the re-keying problem of justice data.
- No ability to examine and improve business process or workflow.
- Software must be built or acquired to transform the many data sources into a useable format.
- There is no event notification when data changes, the requestor must constantly re-query to get the latest information.
- A data warehouse is not integration; it is just part of a data query tool.

Centralized Index

A centralized index works like a telephone book; it only maintains indexes to where the actual data is located. Once the data is located then the operations are similar to the virtual query.

Advantages:

- The native applications are in tight control of their data.

Integration Limitations:

- A centralized index makes two copies of the key (indexing entity); the index must be in constant communication with source systems to ensure the indexes are valid.
- If what you want is not in the central index, you have to submit a development request.
- There is no system-to-system integration, so inconsistent information between systems is still possible.
- Does nothing to solve the re-keying problem of justice data.
- No ability to examine and improve business process or workflow.
- There is no event notification when data changes; the requestor must constantly re-query to get the latest information.
- An end user query tool is still needed; people cannot just access the index to find the data.

Justice Application Suite

All justice applications are from the same vendor, similar to buying all your PC applications from Microsoft.

Advantages

- Everything works together and data integration is built in.

Integration Limitations:

- Buying justice applications as a suite from one vendor is not practical at the state level due to cost, politics, and installed applications. At the county level, suites are a possibility.
- As processes and needs change over time, there will be constant modification of code that may or may not be maintainable.
- Best of breed applications cannot be utilized; you are stuck with the vendors' capabilities.

Enterprise Application Integration (EAI)

Developed in the late 1990's and validated in the private sector, EAI is what technology analysts such as Gartner, Meta Group, and major vendors (Oracle, IBM, SUN, SAP, Peoplesoft…) define as integration. EAI is a set of software tools and design techniques that provide the following:

Advantages

- Data is shared between applications, not from a secondary source; data is current.

- Applications can update other applications when changes occur (based on business rules).
- New applications or application changes can be made and cause minimal to no impact to other systems.
- With data integrated, business process and workflow improvements can now be analyzed and addressed.

Integration Limitations:

- It takes considerable more planning to implement an EAI architecture since data is looked at in an enterprise sense, not just in silos. Once the plan is established, then rolling out an EAI can be done in phases.

4. Enterprise Application Integration (EAI) - Explained

4.1 *Why is EAI different?*

EAI is a data architecture for sharing data to make applications consistent and current in the usage of key business data. EAI is not an application such as a portal, data warehouse, or a consolidating index; these are applications because their purpose is singular in purpose, to consolidate information for a query. The definition of data architecture from Carnegie Mellon Software Engineering Institute helps draw the distinction:

> *Data architecture defines how data is stored, managed, and used in a system. It establishes common guidelines for data operations that make it possible to predict, model, gauge, and control the flow of data in the system. This is even more important when system components are developed by or acquired from different contractors or vendors.* [2]

4.2 *The advantages of an EAI approach to Justice Integration*

Many papers, presentations, and books have been written to answer the question of why use EAI. Not to minimize those efforts, the advantage of EAI over other data integration patterns is simply that EAI provides business flexibility to the applications you have purchased. Applications have been purchased to solve point issue, e.g. computerize criminal history data or an automated calendar system for courts; each of these are point solutions. Just as much as it takes many different types of specialized agencies and their people to provide safety and security to the citizenship, it has also taken specialized applications to assist them in performing those duties. To continue the analogue, no agency can stand alone in providing all the services needed in the justice community. Everyone needs to communicate with each other to get their jobs done. The applications of a justice system need that same level of communication. EAI is to justice applications like the telephone system is to people of the justice community. People can change offices, buildings, and even agencies and you can reach them with a phone call. EAI provides the same level of access to data (following business rules). Once the EAI architecture is in place, adding or changing applications no longer requires many months of software development. No other data integration pattern provides the business flexibility of EAI.

4.3 *Fundamentals of EAI*

4.3.1 The Logical Process of EAI

EAI technology takes application data, normalizes it and then routes the information to the appropriate application, and then converts the normalized data back into application

[2] An Enterprise Information System Data Architecture Guide CMU/SEI-2001-TR-018 ESC-TR-2001-018

specific format. The prior alternative was to write custom interfaces to each application; this old method becomes a maintenance nightmare after about three interfaces.
Figure 1 illustrates the difference between EAI and old interface integration. Advances in EAI technology have added business rules and system management capabilities to the solution stack. These functions will be discussed later.

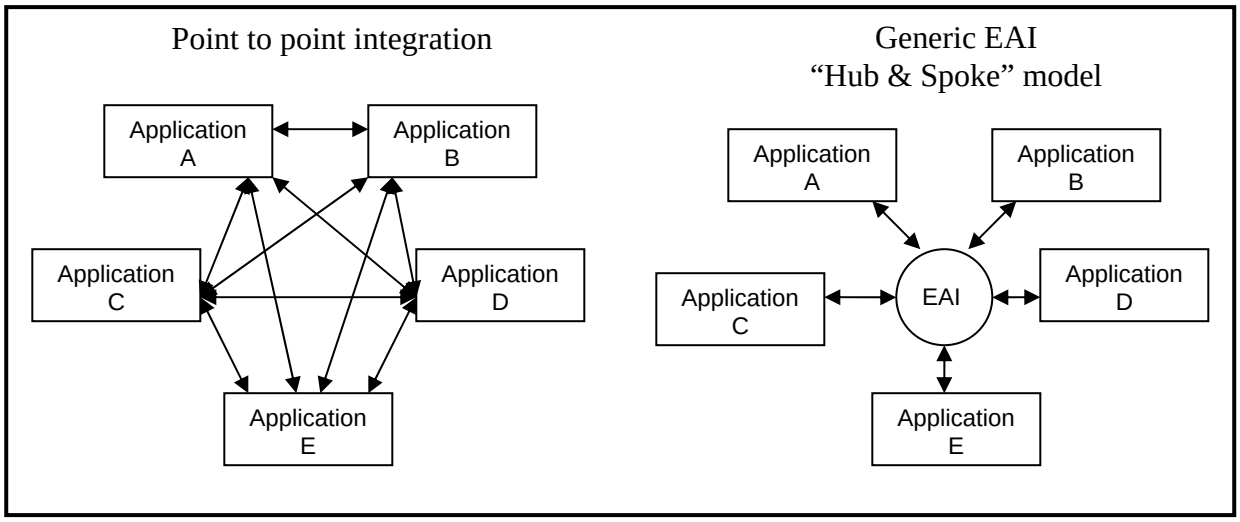

Figure 1

4.3.2 Getting the data for EAI

Applications process data. Data is entered into the application, it is processed, and a result is stored and/or sent to the requestor. This is the generic functionality of any application. Integration requires access to the data of an application to be able to integrate the application with other applications. There are only two ways to access the data in any application:

1) Access the data directly inside the running application.
2) Access the data where the application stores the data (where data rests).

Access the Data in the Application
The most common way of getting inside an application is through an API (application programming interface). APIs provide a model to a programmer of how to write code to communicate to the running application. Getting the data directly out of an application may seem like the best way to get data for integration, but it is not.

Accessing data via an API method is almost never used because:

- Most old applications do not have APIs.
- The APIs are so arcane in usage as to be almost unusable.
- Changes to the application by the vendor (upgrades) may make the API unusable.

Access the data where the application stores the data

Applications store data in flat files and databases. This is the most common place to get application data for integration because:

- How to access flat files or databases is well understood.
- The data elements are stable and not transient as in a running application.

Every vendor of EAI products provides adapters and an adapter's developer's toolkit. Adapters perform the conversion of data into and out of an application. Adapters make accessing most files and databases almost a point and click operation. For those data stores that have no stock adapter, the adapter tool kit can be used to build an adapter to access the data. FIGURE 2 illustrates the EAI components.

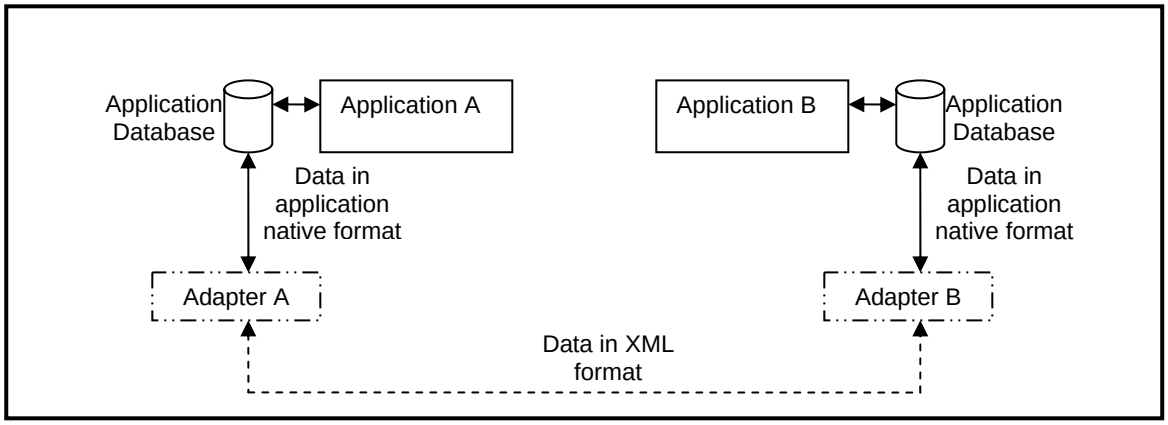

Figure 2

4.3.3 Integrating Application Data

EAI systems can either poll for data to integrate or be sent data by the adapter when it is changed in the application. This is called event-driven data. In either case, the EAI routing engine takes the data and determines what other applications have registered to receive this data set, and sends the data set to those applications. The EAI routing engine can be loaded with rules for data validation, cleansing, and manipulation of any magnitude and scope so that the receiving applications receive the correct data. The capabilities of routing and rules will not be covered in this paper since the capabilities vary by vendor. It is at this point that the data from one application has been integrated to the other application. FIGURE 3 illustrates the EAI components at this point.

FIGURE 3 – take figure #2 and add routing engine

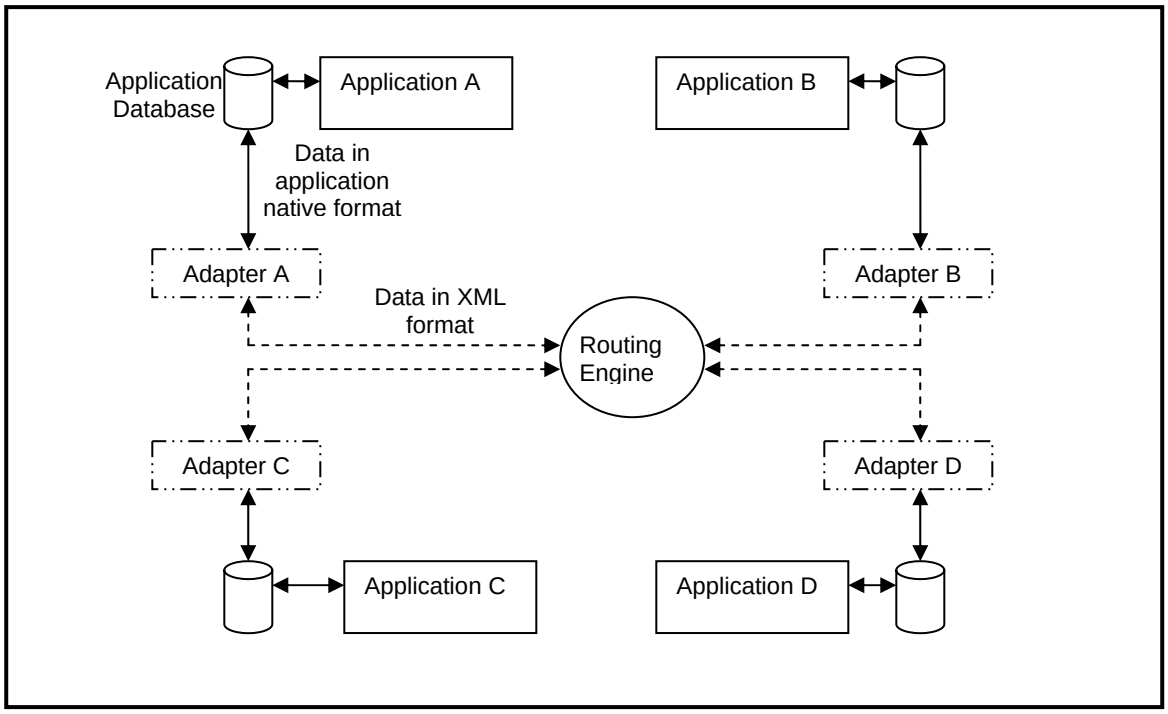

Figure 3

4.3.4 Web Services, XML, and EAI

Many people think that web services and XML obviate the need for EAI. By way of analogue, the world would not need translators if everyone spoke one language. Web services and XML are only a small part of the integration solution. XML is a way to structure a data message, the same as grammar rules structure words on a paper, as to where punctuation occurs and capitalization. XML does not provide the next lower level of organization, knowing that a particular data element is a valid SSN. XML just provides a structure for holding SSN data in a data message. XML is routinely used as the message format in EAI products since the format is well understood and easy to debug.

Web services are a three-part technology that leverages XML. The three parts are:

1) SOAP – Simple Object Access Protocol
   SOAP defines the vocabulary for the message (the envelope and payload format).

2) WSDL – Web Services Description Language
   WSDL is an XML document that describes what a service does and how to communicate with that service (typically the internet protocol, HTTP). When a service is needed, the WSDL request describes what is being searched for and the communications protocol needed to communicate when the service is located. An example of a service would be a mug shot service; pass the mug shot service a SID and receive back the mug shot.

3) UDDI – Universal Directory Description Interface
   UDDI is a "yellow pages" of available services on the network. If a mug shot service is needed, UDDI will tell the requesting service where such a service is located and the interface parameters.

The functionality still missing from web services and XML are:

- Applications still do not speak XML; adapters are still needed.
- There is no validation that the data in an XML packet is valid. The data in the SSN placeholder may not even be SSN or in valid syntax.
- Routing the data message to the right location in a quality of service framework.
- There is no information (in the standards) that tells how the data is to be processed (or validated).
- Security and authorization standards are just starting to be proposed (WS-I Security).

EAI technology supplies all the missing functionality. It should be noted that EAI technology vendors without exception embrace web services and XML. The technology will allow many of the mundane low value tasks (building adapters) to be avoided and it opens a higher value world of business process and workflow management.
FIGURE 4 illustrates the EAI components at this point.

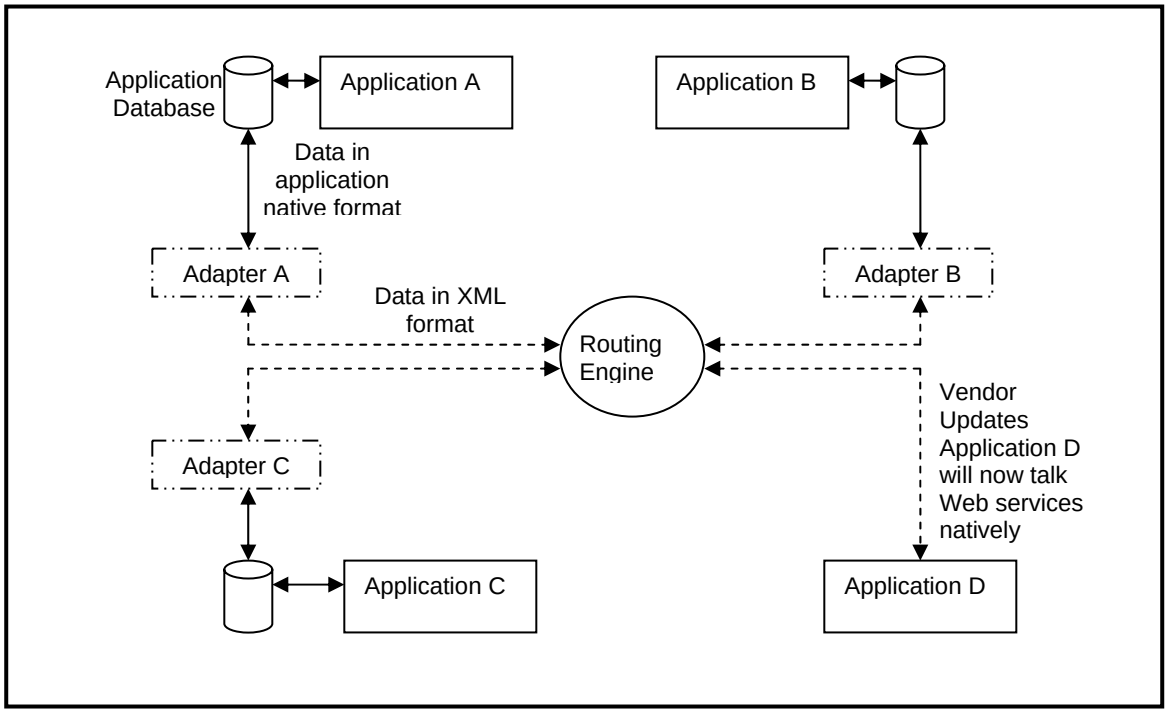

Figure 4

4.3.5 The case of industry specific XML

The power of XML to describe its own internal data has led to the generation of many industry specific XML models. GJXDM (Global Justice XML Data Model) is one of these industry specific XML models. An industry specific XML data model not only

describes the elements of the data model but also the data type (e.g. integer, real, character). For example, the data model for ARREST TYPE in GJXDM has many data elements, some of which are activity ID, date, and time. Each data element has been typed to be of a specific data type. If you use GJXDM and place data in the time field, it will be the same format as anyone else using the GJXDM data type - time. GJXDM (and other industry specific XML formats) are forcing a common defined language. If all justice applications spoke in GJXDM, then there would never be a need to translate (in this case, time) a data instance from the CAD system to the booking system. Even with industry-specific XML models, there will be a need to use translation programs/adapters to translate data between applications and the XML model. In the future, when all applications can interface to GJXDM, then adapters will not be required.

5. EAI to SOA

EAI is a data architecture for integration; Service-Oriented Architecture (or SOA) is a system architecture. A system architecture incorporates data architecture as well as many other architecture designs, management, security, and infrastructure among others. This paper is concerned with the general aspects of how SOA drives integration architecture to benefit justice integration.

Several technology initiatives have combined over the last decade to make SOA the recommended system architecture for integration. Those technologies are XML, web services, and EAI. SOA is a method of building systems; it is not a product. SOA describes how applications are designed, built and interact with each other. An SOA describes an application as a grouping of many services or business processes. A service is analogous to a business process. Validating and retrieving the data for a given SID would be an example of a service or business process.

Applications, and therefore services, are decoupled from each other (abstracted). The data movement is abstracted from the application, and the business process (service) is exposed. While this may sound implausible, it is possible due to the technologies of XML (which provides a standard way to represent data), web services (which provides a standard way to locate and utilize services), and EAI (which provides data transformation, and the abstraction of data from applications).

The concepts of SOA can be applied to legacy applications by wrapping the application in service points. Vendor applications can be treated the same way.

A common world analogue can help explain SOA. The relationship between an electrical appliance and the electric utility is an example of service-oriented architecture. The electric company has no idea how the electric is being used and the appliance does not care how the electricity is supplied to the wall plate. If the appliance and the utility were a tightly-coupled, dependent relationship, then every time you buy an appliance, you would need to call the electric company and register the characteristics of the appliance. Depending on those characteristics, the electric company may have to string another line into your home to service just that appliance. Interface standards and processing standards provide the flexibility in the electric system. The community of justice integration is closer realizing this services model than even the private sector due to the work in XML by GJXDM and modeling efforts by SEARCH and others (URL Integration).

At its core, SOA is about abstracting data, business rules, requestors of service, and consumers of service from each other. The less each of these components knows about the particular implementation of each other, the easier it is for the justice system to handle business process changes. So how does an application work if it is abstracted (hidden) from its data? How does one application communicate with another if they are abstracted? It is not magic. The technology used is in place today, XML, Webservices,

and EAI. The justice communities get an extra boost with the normalized data model GJXDM and the process models provided by SEARCH and others.

5.1 *SOA Operations in Justice Integration*

As much as GJXDM is attempting to model the data entities in a standardized justice environment, the SEARCH process models are attempting to model the business rules in a standardized justice environment. Using these efforts as a starting point, any justice system can be defined with a core set of data entities and a library of business rules that act on these data entities. The term “common business object” best describes these data entities because they are not flat structures (e.g. simple attributes about a person) but are rather very complete descriptions (e.g. person attributes and their criminal history). The common business object for an arrest record, if populated with data, would contain a complete picture of the arrest record for a suspect. Common business objects are specialized containers structured in XML following the examples set forth in GJXDM. The data from legacy systems interacts with common business object through adapters. Newer applications would be written with a web services interface and would not need an adapter.

The business objects are used by services (applications) loading (fulfilling a service request) and offloading (fulfilling a service consumers request) data. How that data is acquired, processed, and secured is detailed in the business rules. Leveraging and building on the work of the SEARCH process models (business rules), the business rules for justice have been abstracted from the data. The where and how of data storage is not a concern of the business rule. The business rules are also described in a form of XML so that are independent of any operating system or programming language; they are abstracted from this level of detail.

The SOA justice system now has data represented as common business objects which are acted on by a library of business rules. The business objects are transported in the system by messaging software that provides the message delivery mechanism, getting the message to the right place, guaranteed delivery, fault tolerance, and security. The messaging software provides the needed integration features not found in TCP/IP and other networking protocols.

Finally, in the SOA justice system, an orchestration engine is needed to apply the right business rules to the right common business objects. The orchestration engine is where business rules and workflows can be defined, modified, observed, logged, and secured. Orchestration engines are very graphical in nature so that business analysts can readily understand the system operations.

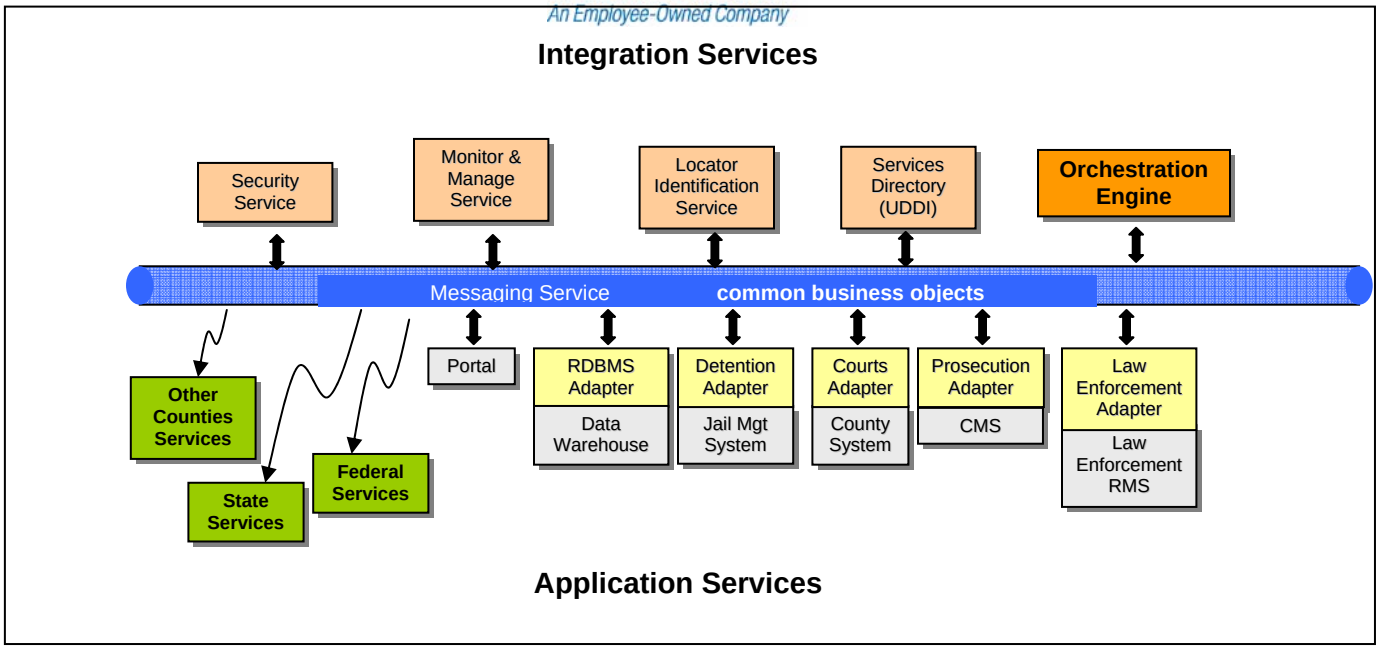

Figure 5

5.2 *The advantages of SOA for Justice Integration*

The full business value of SOA is realized when the applications of the justice environment can start to be viewed and used as services. Instead of being concerned about the where and how to integrate applications, an SOA changes the view to the what and why of how applications (or parts of applications) can be combined (as services) to affect the business process. In an SOA environment, the focus is about what you want to do, not the how. If the need in a business process is to check for a valid SID, then that process (the SID service) can be virtually drag-and-dropped into the process chain to make it part of the business process. There is no need to worry about where the SID validation program is located or how to write code to access the program; all application-specific information has been isolated (abstracted) in the SOA. The focus can now be on what services are needed to improve the business process. Other benefits of an SOA approach to justice integration are:

- New applications can almost be “plugged in”. To bring new applications on-line requires determining which common business objects will be used (or added) and which business rules will be used (or added). No new interfaces need to be written to other applications or data sources, because they are all ready part of the integration fabric. If a justice portal or a data warehouse is to be added to the justice environment, then those applications can be defined as services on the messaging infrastructure. The mapping to the common business objects and business rules these new services need are setup in the orchestration engine. In the future, if the portal needs to render another new data set, then the common business object with that data set is made available to the portal. There is no need to write a data access routine to an application’s database to query the data since the application’s data has already

been made available in a common business object when that application was brought on-line.

- Business processes and workflows can be analyzed, modified and measured. SOA catalogs the business processes and business rules. SOA provides that common language architecture for data business processes that allows the manipulation of business processes across the entire justice system if that is what is desired. With SOA, justice practitioners can now truly work on process improvements.

- With easy access to process and workflows, legislative changes can now be processed and the impact can be measured and assessed.

- Security and authorization can now be monitored at the business process level, not just at the data level.

- SOA provides the architecture for developers to reuse rather than build new. The more applications that are enumerated as services, the bigger the library of services becomes. The more services available, the less development time, effort, and cost are required for new “applications”. Developers can build new applications (services) faster and maintenance becomes less a factor since the majority of the services used will all ready be well formed and tested.

6. Conclusion

Integration of information is a critical part of the justice information environment. Getting the right data and a complete picture of the requested information when it is needed is a critical need in servicing the justice community.

Many different data integration patterns have been tried over the years and the evolution has lead to a system view of the problem rather than just a data view. The service-oriented architecture (SOA) that has been proven in the private sector provides the capabilities needed in justice integration. These capabilities include the ability to integrate large numbers of disparate systems, maintain control on data and business processes, and provide the ability to improve, modify, and measure process and workflow activities.

There is hard work in implementing an SOA, and that work focuses on the cooperation between agencies and personnel. The quality and benefits of the SOA are directly related to the completeness of the common business object definitions and business rules. The process models of SEARCH and the XML library of GJXDM go a long way to making an SOA achievable. However, to make an SOA work in a particular justice environment will take participation and validation from the justice practitioners. Only then will the full value of implementing SOA to solve the justice information integration problem be realized.

7. About the Author

Mr. Lengerich is the system architect for the Integrated Justice Information Solution from Science Applications International Corporation (SAIC). He has 23 years of IT experience. Since 1990, the author has been involved in infrastructure technology companies in both database (Sybase) and integration middleware technologies (TIBCO Inc.). While at TIBCO, he established and managed the competitive intelligence department. In this capacity, he was involved in product strategy & development, mergers and acquisitions, and sales execution. The author has presented on the subject of product strategy and integration solutions at conferences and user events worldwide.

Mr. Lengerich has a BA in Biochemistry from the University of San Diego and an MBA in Information System from Regis University, Denver. He has been an adjunct teaching faculty member of the Computer Science Department at Regis University since 1990.

The author is a member of the IJIS Institute XML Committee.