MTR070065

MITRE TECHNICAL REPORT

Practical Words about Shared Service Identification

April 2007

Stacey Darnell

Sponsor:

Dept. No.:	E142	**Project No.:**	03075620-XA

MITRE

Approved by:

Michael McFarren
Department Head, E142

Practical Words about Shared Service Identification

Stacey Darnell
The MITRE Corporation
McLean, VA

ABSTRACT

This document describes a path to identify shared services while maintaining the connection with the business need. The business data fields included in the process outcomes are defined by the business team. The business process outcomes are evaluated and graduated as services for automation. Technical information is collected about each graduated service. Technical flows are created to identify the technical components and how technical components are used to support the service. Services are identified and service metadata specified.

SERVICE DEFINITION

This paper will take direction and definition from the Organization for the Advancement of Structured Information Standards (OASIS).[1]OASIS refers to a service as:

- The capability to perform work for another
- The specification of the work offered for another
- The offer to perform work for another

OASIS continues by saying, “Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains … The perceived value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs.… in SOA, services are the mechanism by which needs and capabilities are brought together. SOA is a means of organizing solutions that promote reuse, growth, and interoperability.” SOA is an “organizing and delivery paradigm that enables one to get more value from use of both capabilities which are locally owned and those under the control of others.”[1]

This paper will maintain that a need without a capability is limited, and a capability without a need lacks purpose. The need is usually aligned with the business aspects of SOA and the capability is usually facilitated by the technical aspects of SOA. For this reason we start the search for services within the outcomes of the business processes. The business process is in place to provide for a business need. This paper takes place at a point where business input is available.

NOTES ABOUT PROCESS OUTCOMES

Processes are sometimes created by the professionals who perform the tasks but not by business analysts with training in proper process capture. Further, the business models do not always align with technical models that implement them.[2] The information in the process diagrams is sometimes only understood by the audience who created them. The processes would not be understood by other audiences or by implementers asked to automate them. This lack of understanding can result in loss of time, money, talent, and opportunity. However, even these processes can be helped with a few simple techniques.

At a minimum, useful processes have a visible hierarchy, all the inputs and outputs evident, and clear step sequence. If processes are not in a useful state then bringing them to a useful stage is the next step. The process outcomes are very important for service identification.

Documents in the Chain to Reality series under the “Raw Process Capture” link goes into greater detail about making processes *useful* and then making processes *strategic*. [3]For now, this paper begins with the assumption that the business processes are at least in a useful condition. We start at the point where we have collected all of the outcomes from our processes into a list. Not all process outcomes require automation to provide a service.

OVERVIEW OF STEPS TO IDENTIFY SERVICES

1. Evaluate Outcomes: The next step is to evaluate which process outcomes should be shared with the help of technology. Evaluation criteria (discussed later in this section) are necessary to judge the “service worthiness” of an outcome. Outcomes that are deemed service worthy are listed as outcomes that could move forward as automated services. The reasons for not moving forward into automation on a particular outcome should be documented.

2. Collect Targeted Technical Information: Before we can understand what it will take to offer the service, we will need to engage with the technical team. We will collect the necessary information about the sources needed to support the outcome. Pertinent technical information is collected

so expectations can be set about the system's capacity to expose and provide the data for the service consistently.

3. Create Technical Flows: Once the source system's capacity is understood, a **technical flow** is created. A technical flow is a sequence of steps showing how data flows between technical components that support a service offering. Technical flows are created for all process outcomes that need to be shared and deemed service worthy.

4. Identify Services: The technical flows are examined to identify where a "read" or "write" event occurs. For instance, some information can only be viewed. Some information can be written to, and some service offerings allow both a "read" and a "write" event. Finding out what can be done to the service establishes what the **data authority** (create, read, update, delete) is for that service.

5. Specify Metadata: Compile all of the "read/write" events and create a list of services to support the process outcome. Make sure to include any "help" functions that are presented universally across all process outcomes. Document the **metadata** for the services. The data elements of the data bearing process outcome are used to supply the metadata for the service. Document any supporting services that facilitate a service flow. For instance, a service solely dedicated to making a request would be a supporting service.

1. EVALUATE OUTCOMES

After the process outcomes have been listed they can be evaluated for service worthiness. Not every process outcome is service worthy for automation. For instance, a training process may result in a trained team. A trained team is the outcome that meets the goal of the process. This is a good outcome, but not one that becomes an automated service.

The most desirable service worthy candidates are usually "data bearing" candidates. Part of understanding what to share is based on common sense. However, there are two other aspects to consider for service worthiness.

- The business case
- Service Characteristics

1.1 Building the Business Case

Deciding if an outcome should become a service is based on value. First, does it even make business sense to create a service out of the process outcome? If the process outcome provides no value as a service, then it should be eliminated from the list of service possibilities.

An example will be used through the remainder of the paper to illustrate the concepts presented. The example takes place within a response center that manages different types of incidents related to the wellness of technologies in production, and the customer experience of those using the technologies. The incidents are recorded and then assigned to resolution divisions with the expertise to solve the incident. The response center generates a reason code list that can change at any time depending on what is happening in real time within the organization. The reason codes are critical for generating response tickets that are given to the correct responders. In the example, the business team seeks to build a case to offer the current reason code list as a service. The reason code list would be used to automatically generate incident tickets. Suppose the notes below were taken during a meeting with the business team.

Table 1 Building the Business Case

Category	Sample Response
Shared Service Business Just-ification	Business Goal: Automatic Ticket Generation from within Enterprise Systems for Response Center incidents. Automate: The Response Center reason code availability is essential to the creation of automated tickets. Dealing with Response Center reason code combinations is a manual process today for downstream consumers of Response Center incidents. The manual nature of today's reason code distribution limits the Response Center to automated benefit of only a few specific events that are automated through hard coding. Maintaining the few events available has proven to be costly. However, as a service, newer technology would expose all the incident events electronically available for ticketing purposes and in real time. Ticketing information is gathered by operating from a static list and through re-keying the incident data 3-5 times within the course of an incident. Team members are not operating from the same or most current list of reason codes. The current

Category	Sample Response
	distribution of the list results in a host of costly errors. Reasons for Sharing: 1. Increase Accuracy The same and most current list will be available to all team members. Re-keying would be unnecessary thus reducing the point where errors can occur. 2. Enhance User Experience User experience benefits are also sought within the Response Center. Even with the dashboard, a Response Center user still has 10 + windows open. This automation could also serve as an application reducer on the user screen. 3. Increase Scalability The automation of all Response Center reason codes will help the Response Center scale up to all event types.
Impacts from a Team Perspective	The Response Center team can grow across more Response Center reason combinations and offer them automatically to downstream systems. • Reduce windows open on the Response Center user workstation. • Automate combination comparison of Response Center Reason codes. • Cut waste out of the process overall. Downstream teams can eliminate 3-5 re-keying processes per incident, make assignments quicker and based on skills, reduce errors from static Response Center reason code lists and re-keying, and give faster attention to high priority issues.
Impacts from a financial perspective	Financial gains stem from an estimated three minutes saved per incident due to improved routing. This benefit benchmark does not include improvements in the downstream teams. Each minute of an incident costs $X.XX. There are 20,000 incidents coming into the Response Center per month. This simple change translates into $XX.XXX in savings per month for the Response Center alone.
	Productivity gains are also expected from the time an incident occurs to implementing the solution. There are also saving in licensing charges. The licenses necessary to maintain come down in price when redundancies are removed from the ticket resolution process.
Visions of solving the problem with a shared service	A shared service can be accessed by any downstream partner of the Response Center. The service shows the most current code in real time and can be consumed for processing. Every downstream team would become a partner in the Shared Services Environment where the service is offered. The partnering teams can integrate the service with the reason code data directly into their own business process to ensure a quicker time to solution. The partnering teams of the Shared Services Environment that receive the reason code updates can do so automatically (machine to machine) or through a portal for user consumption. Every team will be operating from the most current information available.
Impacts on growth	The same team member can process more tickets. This output gain is important as new responsibilities are added with the merging and reorganization of operational units.

At the end of the business case evaluation several key answers can be given.

Business Information:

- What is the business case and intention for the service?
- Who are the business stewards who own the processes that create the outcome?
- What is the proposed use for the service?
- What is the expected benefit?

The business team makes a strong case for the automatic ticketing endeavor shown in Table 1. However, the case from the business perspective is not the only criteria to consider.

1.2 Service Characteristics
The evaluation is taken to a deeper level by looking at what is characteristic of shared services.

If an outcome is to be a good candidate for a shared service, then the outcome should emulate the characteristics of a shared service. You may have heard the saying, "If it walks like a duck, and talks like a duck, then it's a duck." In our case the saying would be, "If it acts like a service, and shares like a service, then it's a service."

Shared characteristics and their opposing characteristics are presented for evaluation.

Shared Characteristics:

1. Consumes existing outcomes
   - Opposite characteristic: 1. Builds data output to be consumed
2. Workflow is accomplished by connecting shared services together
   - Opposite characteristic: 2. Workflow is accomplished by using the source system's internal workflow engine
3. The service can mix and connect with other services without needing to know where the other services originally come from.
   - Opposite characteristic: 3. Sharing data outside the source system is accomplished with specific instructions and interfaces to other systems.
4. The service is made to be shared across machines and boundaries and as a part of high volume server applications.
   - Opposite Characteristic: 4. The data and processing of the data is local to specific machines and can only be shared across machines with specific instructions. Sharing across machines is accomplished with difficulty.
5. Services interoperate together.
   - Opposite Characteristic: 5. Systems integrate with each other.
6. Standard driven technology. The standards are focused on interoperation.
   - Opposite Characteristic: 6. Standard non-specific technology. Standards exist but are proprietary to the system. Ad hoc compliance with industry interoperation standards.
7. Interaction is more dynamic.
   - Opposite Characteristic: 7. Interaction is more static. A dynamic result is achieved with difficulty and cost.
8. Services are delivered for incremental use and in an on-going fashion like an assembly line. Typically services are delivered every 30 days
   - Opposite Characteristic: 8. Applications are delivered by isolated teams in response to a project charge. Applications are delivered every six months to a year.

The Point: If an outcome of a process such as, reason code list, can match up in behavior with the service characteristics then the outcome is a strong candidate for an automated service.

2. COLLECTING TECHNICAL INFORMATION

The purpose of collecting technical information is to determine the capability of the technical source. The technical source may be an older system, but remains valuable. The technical source may require other technical components to assist with transporting and transforming information to the Shared Services Environment. How the service is exposed is important information for setting expectations on how the service will perform.

Technical information gathering starts with what was provided by the business team. The business team defines the data elements that must be present in process outcomes to perform their work. (See documents in the Chain to Reality series under Process Modularization link)[3]

The technical team maps the technical fields to the business data elements specified in the process outcome.

Technical Source Information for the field mappings include:

1. The source system name(s) that support the outcome
2. The name of the system owner/data steward for the source system
3. The business and technical fields that make up the consistent field structure of the outcome
4. The table, field, size, and type information

Information beyond the above list is gathered if an outcome is graduated to a service. Additional information is gathered to fully describe the process outcome. For more definition on service description consult the Automation package link and the Runtime Environment link of the

Chain to Reality series.[3] Guide questions are available to help the team member collect targeted technical information. For now the major categories are given for data collection.

- Service Definition
- Service Consumption
- Service Exposure
- Service Meta Data
- Service SLA's

The information collected is refined following the discoveries that are made with technical flows. The outcomes from the processes are still in a stage where they are "elected" as services, but not completely defined. The engagement relationships between the provider, shared services environment, and the consumer can be seen in the service description information.

3. CREATE TECHNICAL FLOWS

Technical flows describe the technology support needed for the service to function. Technical flows can be scripted and diagrammed. Information from the business process flows and the technical information that has been collected during the technical interviews serve as input for creating technical flows.

First, revisit the process flows:
Review the processes and look for every occurrence of the process outcome. Notice how the process outcome is being used in the processes. Pay close attention to places where the process wants the user to supply information, read output, or look up information.

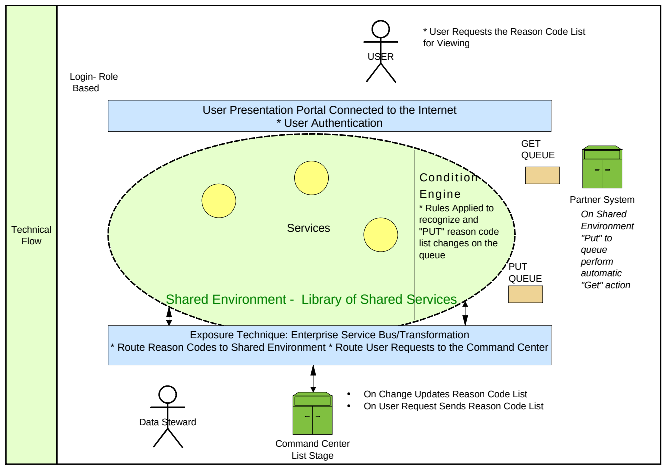
For example, are there any points in a process where a user searches for a reason code? Are the reason codes always generated and distributed as read-only?

Second, revisit the technical information:
The technical information will determine the technical utilities necessary to support the technical flow.

Diagram the technical utilities. Figure 1 shows the technical tools that will be used to facilitate the technical flows for the services supporting the process outcome "Reason Code List."

From our research we have identified that the technical flow for the "Reason Code List" will leverage older and newer technology:

- A user
- A data steward
- A user presentation portal
- A Shared Services Environment
- A Shared Services Environment condition engine
- Services
- Enterprise service bus
- Source System
- Partner systems
- Queues for guaranteed delivery



Technical Flow
USER
* User Requests the Reason Code List for Viewing
Login- Role Based
User Presentation Portal Connected to the Internet
* User Authentication
Services
Condition Engine
* Rules Applied to recognize and "PUT" reason code list changes on the queue
GET QUEUE
PUT QUEUE
Partner System
On Shared Environment "Put" to queue perform automatic "Get" action
Shared Environment - Library of Shared Services
Exposure Technique: Enterprise Service Bus/Transformation
* Route Reason Codes to Shared Environment * Route User Requests to the Command Center
Data Steward
Command Center List Stage
On Change Updates Reason Code List
On User Request Sends Reason Code List

Figure 1 Components for Technical flow

Although the flow lines in the picture have not been drawn the sketch is a good start toward documenting the components used from the Shared Services Environment and the source system to accomplish the technical flow.

<u>3.1 Narrate Technical Flows</u>
Our research has produced two scenarios for the use of the reason code list.

Scenario 1: Automatic updates for reason codes
Scenario 2: User views Reason Code list from portal

The steps for each scenario are written down in story line resembling a use case.

<u>Scenario 1: Automatic Updates for Reason Codes</u>

1. Response Center reason code change occurs in Response Center System
2. On change, XML Message prepared with Reason Code List Changes
3. XML Message placed on Enterprise Service Bus (ESB)
4. ESB transports updated reason code list entries to Shared Services Environment….etc

<u>Scenario 2: User Views Reason Code List from portal</u>

1. The user authenticates via a login into a presentation portal based on the Shared Services Environment
2. The user invokes the “See Reason Code List” choice in the user portal
3. A request for the list is routed to the ESB for delivery
4. The ESB delivers the request to the Response Center System
5. The Response Center System processes the request…etc

4. IDENTIFY SERVICES

Look carefully at the technical flows. In particular, look for occasions when a “read” or “write” event occurs.

Observations are made in regards to the “Reason Code List” example. The technical flows reveal what is happening to the reason code list.

Observations made across both scenarios:

- Conclusion: Data authority for the reason code list is read-only.
    - Reasoning: Across both scenarios, once the reason code list is offered, no user or machine can write to the list. Nothing is allowed to change the values of the list. The list is “READ ONLY.”
- Conclusion: The only time a user performs a “write” in the flow is when a request for the reason code list is made from the portal.
    - Reasoning: The user portal scenario allows a user to ask to see the reason code list. When the user selects the option to see the list, the user is performing a “write” operation that populates the request for the list.
- Conclusion: The reason code list is the same structure regardless of the scenario
    - Reasoning: For both scenarios, the reason code list is sent from the Response Center System. The same data elements for the read-only list are sent regardless of a request from a user display or an automatic update. The message is exactly the same in both cases.
- Conclusion: Delivery must be specified.
    - Reasoning: The delivery is either to a portal for user consumption or to a queue for machine consumption.
    - The condition engine in the Shared Services Environment must be able to tell the difference between a message coming in for updates versus a message that displays reason codes to a user on the portal.

The observations from the technical flows show the services necessary to accomplish the flow. There must be a service to invoke a request from a user. There must also be a service to present the reason code list. The reason code list is the same data structure for both scenarios. Only one service is needed to represent the reason code list. The list of services for the “REASON CODE LIST” outcome is as follows:

1. Request Reason Code List: Service – Write –One Direction
2. Present Reason Codes: Service–Read –One Direction

Figure 2 illustrates the interaction of services with the technical components engaged from the source system and the shared service environment.

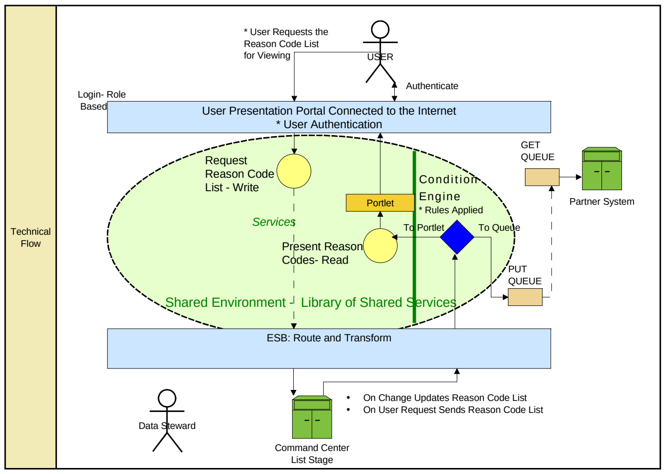

Technical Flow
* User Requests the Reason Code List for Viewing
USER
Authenticate
Login- Role Based
User Presentation Portal Connected to the Internet
* User Authentication
Request Reason Code List - Write
Services
Portlet
Condition Engine
* Rules Applied
To Portlet
To Queue
Present Reason Codes- Read
GET QUEUE
Partner System
PUT QUEUE
Shared Environment - Library of Shared Services
ESB: Route and Transform
Data Steward
Command Center List Stage
On Change Updates Reason Code List
On User Request Sends Reason Code List

Figure 2 Services in Technical Flow

The picture looks more complete now that the services are present. We can see the user make a request and the request handed off to the Response Center System by the ESB. The ESB delivers the reason code list to the Shared Services Environment. The condition engine in the Shared Services Environment decides if the reason code list needs to go to the queue for updates or to the portal for display.

5. SPECIFY METADATA

The Metadata provides the data structure for the service. The elements identified in the outcome will be used to populate the Present Reason Code service.

Reminders:

- Sometimes, additional fields, which may help the technical flow of the service, are not present in the process outcome metadata, but will be present in the service metadata.

- Also, remember that sometimes other services are necessary to supplement the technical flow.
    - We have this case in our example. We need a service to carry a request for the list. The request service does not have anything to do with the elements in the process outcome. However, to achieve the technical flow, we need the request service to invoke the Response Center to send the list.
    - The technical information gathered is refined to document the need for support services.

The process outcome for the "REASON CODE LIST" specified both business and technical elements. For our reference, Table 2 shows the data elements in the process outcome for, "REASON CODE LIST"

Table 2 Reason Code List Outcome fields

Business Element	Technical Element	Other Technical Detail
Reason Code	RSN_ID	Source, Table, Field Size…
Reason Code Description	RSN_DESC	Source, Table, Field Size…
Last Updated	RSN_TME_DTE_LAST	Source, Table, Field Size…
Incident Type	RSN_TYPE	Source, Table, Field Size…

Remember, two services have been identified to specify metadata in support of the "REASON CODE LIST" process outcome:

- Request Reason Code List
- Present Reason Code

Begin by specifying the meta data for the service "Request Reason Code List."

The metadata for the service "Request Reason Code List" does not exist in the process outcome. The request data present in this service is used to invoke the list provider to send the list. The request service has only two fields specified. A time stamp is a standard system field that can be given at the moment the user makes the request. The report code is the field that is used to trigger the preparation of the list.

The fields for the request reason code list service are shown in Table 3.

Table 3 Service Meta - Request Reason Code List

Business Element	Technical Element	Other Technical Detail
Time of Request	TMESTAMP	Source, Table, Field Size…
Report Code	RPT_CODE	Source, Table, Field Size…

For the second service, present reason code list, all the original data fields were used as specified in the process outcome. However, a delivery parameter was added to the service that instructs the condition engine in the Shared Services Environment.

CONCLUSION

Very briefly this document has shown a path to identify services starting with the outcomes from business processes. The business data fields included in the process outcomes are defined by the business team. The business process outcomes are evaluated and graduated as services for automation. Technical information is collected about each graduated service. Technical flows are created to identify the technical components and how technical components are used to support the service. Services are identified and service meta data specified.

Techniques are in place to help apply effort in the most efficient manner. However, techniques do not replace the thinking power of the human.

Whether describing a need, capturing a business process for the first time or evaluating outcomes for services the human should not cease to apply their mental gifts to the task.

More practical information in regards to Service Oriented Architecture can be found in the Chain to Reality series and other referenced documents.

123

[1] OASIS, Reference Model for Service Oriented Architecture, Committee Draft 1.0, 7 February 2006, Document Identifier: wd_soa_rm-cd1, Location: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[2] "From Business Process Model to Consistent Implementation: A case for formal verification models" Jana Koehler, Giuliano Tirenni, Santhosh Kumaran, IBM

[3] The Chain to Reality Series is a collection of practical papers focusing on the essentials for connecting "needs" with "capabilities." The techniques in the series are effective for SOA endeavors –Author: Stacey Darnell