

HUMAN SERVICES

INTEGRATING USER INTERFACES INTO A SERVICE ORIENTED ARCHITECTURE

JEFFREY RICKER
NOVEMBER 7, 2006
VERSION 1.1

ABSTRACT

As unfashionable as it may be to point out, there are still a great number of critical processes in today's businesses that are not automated. How do we create a complete enterprise service-oriented architecture when many of the services still require human interaction or even human execution? Human Services are a pattern for connecting user interfaces into a SOA. This pattern consists of six major components that interact through a process of five major steps. Human services are dependent upon ASAP, a web services protocol that overcomes the "instant gratification" barrier. Through the use of Human Services, we are able to connect non-automated business processes into our overall SOA, which provides the means for rapid implementation and the pathway to future automation.

HUMAN SERVICES

INTEGRATING USER INTERFACES INTO A SERVICE-ORIENTED ARCHITECTURE

CHALLENGE

Service-oriented architecture (SOA) provides a breakthrough in enterprise and trans-enterprise integration. SOA has so far been focused on application-to-application integration and, as such, has lacked a certain human element. As efforts in business process orchestration and enterprise service busses expand, implementers are facing the question: what if there is no application on the other side to connect to?

As unfashionable as it may be to point out, there are still a great number of critical processes in today's businesses that are not automated. How do we create a complete enterprise service-oriented architecture when many of the services still require human interaction or even human execution? Must we automate every service before achieving results? Is SOA an all-or-nothing proposition like so many failed technology efforts of our past? If so, SOA becomes a burden more than a boon.

What we need is a simple, standard means of integrating into an SOA those business processes that still require human interaction, approval or execution. I call this means *Human Services*. We must be able to treat Human Services just like other services so that they are interchangeable with other services. The interchangeability not only enables us to combine such manual processes with applications but also allows us to replace the manual processes with an application in the future with no disruption to the business process.

Human Services must not dictate a particular user interface implementation. To be successful we cannot insist that Human Services user interfaces must be HTML or Java or .NET.

How are Human Services different than existing business process orchestration? Current business process orchestration efforts, most notably Business Process Execution Language (BPEL), do not address human interaction, let alone user interface. Placing humans into an automated business process raises all kinds of messy issues such as:

- Assignment of tasks to individuals
- Sign-on and verification of individuals
- Notification to individuals that they have a task
- User interface for entering the data

These messy issues are why workflow and process flow have been in different realms.

Am I just describing workflow by another name? I can answer that "yes and no." Yes, Human Services achieve much if not all of the objectives of workflow. However, Human Services are different from the workflow reference model published by the Workflow Management Coalition (WfMC). The WfMC developed its reference model as a client-server architecture. It pre-dates SOA. Human Services most closely equate with WfMC Interface 2 and 3 which was last updated in July 1998.

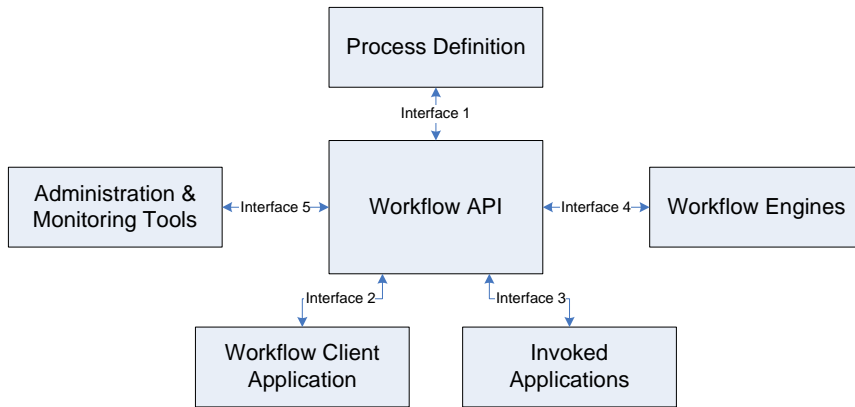


Figure 1 WfMC reference model

The objective of Human Services is to enable arbitrary user interfaces to plug into a SOA. To that end, it must address the messy human things of workflow, but the resulting architecture is as different from the WfMC workflow reference model as client-server architecture is from SOA.

COMPONENTS

A Human Service is comprised of six major components:

- A. ASAP factory service
- B. ASAP instance service
- C. Assignment service
- D. Worklist service
- E. Notification client
- F. Data Entry client

The execution of a Human Service has five major steps:

- 1) Instantiation
- 2) Assignment
- 3) Retrieval
- 4) Data entry
- 5) Completion

The overall relationship between these components and steps is shown in Figure 2. We will discuss each of these major components and major steps in sequence.

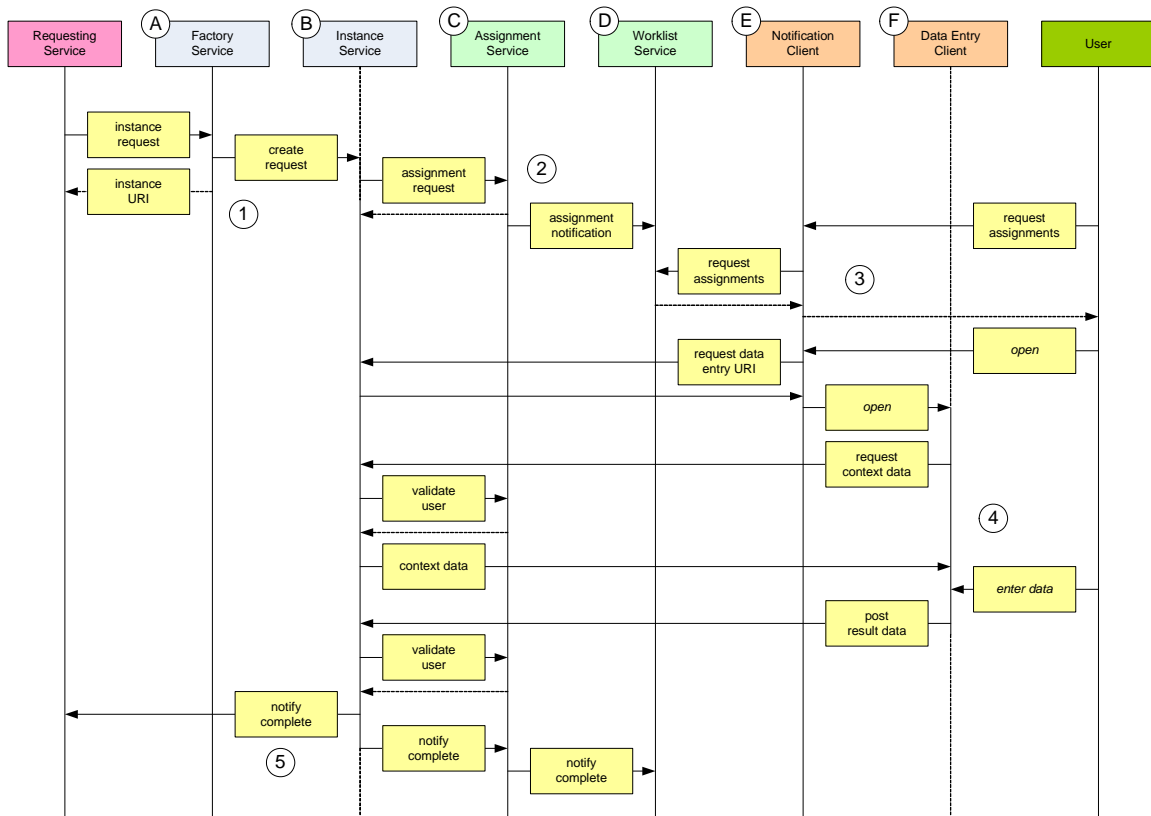


Figure 2 Overall pattern of Human Services

THE FACTORY SERVICE AND INSTANCE SERVICE

Asynchronous Service Access Protocol (ASAP) makes Human Services feasible. ASAP is designed to overcome the instant gratification barrier of HTTP. Normal web services must respond within 60 seconds, or the HTTP connection times out. Many services, such as data mining and chained services, require more than 60 seconds to generate a response. Each service in a chain may individually take less than 60 seconds, but when the services are chained together they the time limit. Most certainly, anything involving a human decision will take more than 60 seconds, as anyone who has stood in line at a coffee shop can attest.

ASAP allows a service to respond to a request, “Where would you like me to send the result when I am done?” There are three types of endpoints to an ASAP service: a Factory service, an Instance service and an Observer service. The requestor sends a request to a Factory service. The Factory service responds with the URI of the Instance service. If the requestor implements the Observer endpoint type (otherwise known as an *interface* in object oriented parlance), then it can add itself as an observer to the Instance service to receive notification of any change of state, most notably completion. When the Instance service is complete, it sends the results to the Observer service. If the requestor cannot implement the Observer endpoint type, then the requestor can continually poll the Instance service for its state.

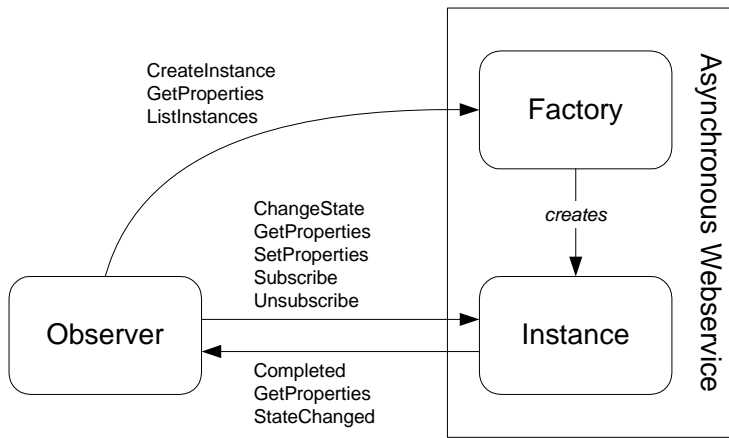


Figure 3 The ASAP endpoint types

A Human Service is an ASAP service. As such, a Human Service has as its first two components a Factory service and an Instance service. What make it *human* are the other four components.

ASSIGNMENT SERVICE AND WORKLIST SERVICE

An Instance service requests an assignment of the task to an individual from an Assignment service. The Assignment service can be very simple and assign all instances of a particular Factory service to a single individual, or it can be complex and balance the assignment of tasks across a pool of individuals. To the Instance service, there is no difference. The Instance service submits its URI to a well known Assignment service and the Assignment service acknowledges the request. By *well known*, I mean that the association between the Instance service and the Assignment service is specified either in the code of the Instance service or in the Instance service deployment environment. Note that the Assignment service implements the ASAP Observer endpoint so that it receives notification of the completion.

The Assignment service notifies a Worklist service of its assignment decision with the URI of the Instance service. The implementation of the Worklist service can be very simple or very complicated, depending on the needs of the deployment, but the interface itself is very simple.

The Assignment service and Worklist service form a decoupling of the user interface from the service. This decoupling even enables multiple user interfaces to be used for the same Human Service.

NOTIFICATION CLIENT AND DATA ENTRY CLIENT

The last two components of the Human Service comprise the user interface. The Notification client is any user interface that notifies the human user that he or she has a task to complete. For instance, the Notification client could be a POP email application, an XMPP instant messaging client or a Java portlet. The Human Service does not care so long as the Notification client can communicate with the Worklist service endpoint.

The Notification client has only the URI of the Instance service. It is up to us how we build the connection between the Notification client and the Data Entry client. We may have the notification and data entry inside a Java-based web portal, or we may have it as a simple LAMP application, or

maybe we build it all as an Eclipse Rich Client Platform. In fact, we could build all three implementations for the same task if we so desired.

The Data Entry client is really where the rubber meets the road. It is the objective of Human Services for us to integrate this user interface into the SOA. Some readers might be frustrated by how openly I have defined the user interface implementation. The point is that the user interface is now decoupled from the service. Using the Human Service pattern we can put any user interface into a SOA so long as it conforms with the basic endpoints specifications.

INSTANTIATION

We have discussed the six major components of a Human Service. Now we will describe the process by which these components interact in five major steps.

The Instantiation step is pure ASAP. The Requesting service sends a request to the Factory service. The Factory service creates an Instance service and responds to the Requesting service with the URI of the Instance. If the Requesting service implements the Observer endpoint type, then it would add itself as an observer to the Instance service.

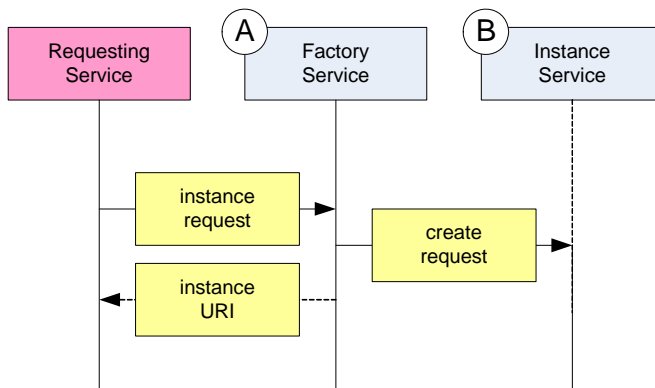


Figure 4 Instantiation step

ASSIGNMENT

In the Assignment step, the Instance service sends an assignment request to the Assignment service. The correlation between the Instance service and Assignment service is completely arbitrary. That is to say, the implementer decides what Assignment service the Instance service calls. It may be hard coded in the Instance service or perhaps determined by the Instance service container.

The Assignment service acknowledges the request and sends an assignment notification to the appropriate Worklist service. The Assignment service contains the internal logic for which Worklist service to assign the Instance service. The Assignment service can be as complex or simple as the implementer needs.

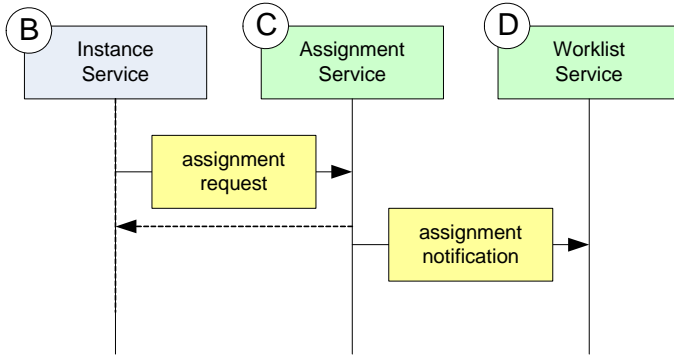


Figure 5 Assignment step

RETRIEVAL

The human user learns what tasks he must complete through the Retrieval process. The user connects to his Notification client which in turn connects to the Worklist service to retrieve the list of Instance service URIs. The interaction between the Notification client and the Worklist service is a simple SOAP exchange. The protocol and user interface technology between the user and the Notification client, however, can be anything and is completely independent of the Human Service pattern.

The user requests a particular item on his worklist. The notification client fetches the list of data entry URIs, picks the most appropriate one and forwards it to the user.

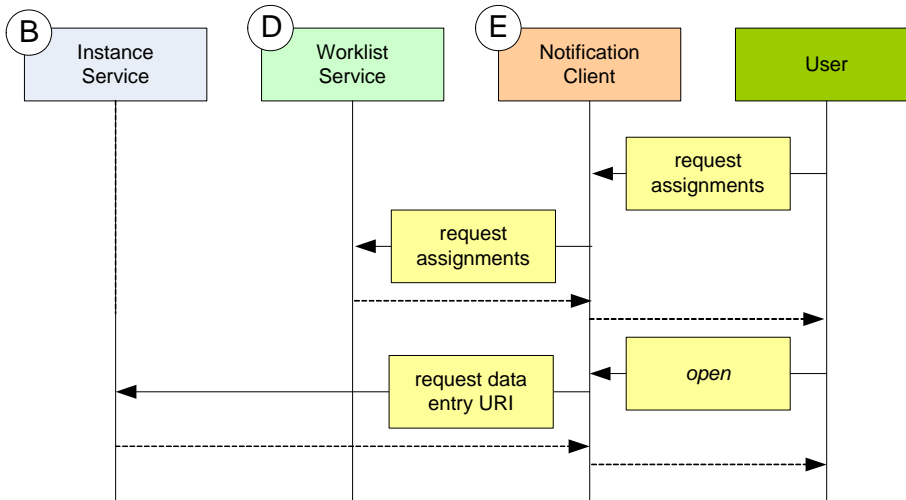


Figure 6 Retrieval step

DATA ENTRY

The next step is Data Entry. Using only an Instance service URI from the Notification client, the user should be able to open a Data Entry client. The specific means of translating a data entry URI into the appropriate Data Entry client is completely up to the implementation of the user interface. For example, an AJAX application will execute this important step completely different than a Visual Basic application.

The Data Entry client sends the request for the context data to the Instance service. The request from the client includes credentials of identification. It may be a simple user name login or it may be a public key exchange. The actual validation is up to the implementation. The critical point, however, is that the Instance service does not handle the validation itself. Instance service asks the Assignment service to validate the credentials of the user. If the Assignment service approves the credentials, then the Instance service sends the initial data to the Data Entry client.

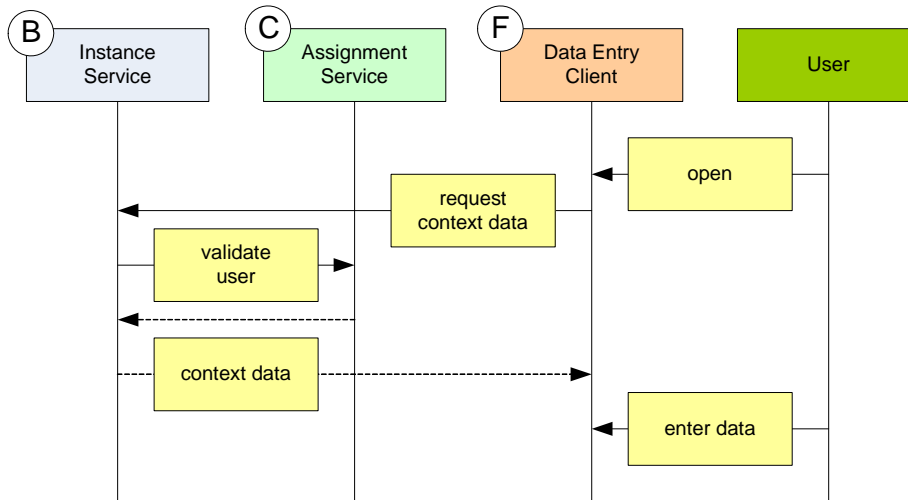


Figure 7 Data entry step

COMPLETION

When the user has completed entering the data, the Data Entry client submits the result data to the Instance service. Note that with ASAP and Human Services, the context data and the result data is XML. That XML could easily be a Service Data Object (SDO).

The Instance service once again validates the user with the Assignment service. If okay, the Instance service notifies all of its observers that the service is complete. In ASAP, such a notification includes the result data, which in our case would be the data created by the Data Entry client. The Requesting service and the Assignment service should be observers to the Instance service. The very last step is for the Assignment service to notify the Worklist service to remove the URI from its lists of tasks.

