OASIS WSIA Technical Committee

Requirements Document
Use Case Report: Customized Producer

Version <1.0>

Revision History

Date	Version	Description	Author
26/Feb/2002	1.0	Customized Producer	Dan Gisolfi, Graeme Riddell, Alan Kropp, Eilon Reshef, Gil Tayar, Rex Brooks, Ravi Konuru, Keven Brinkley, Aditi Karandikar, Monica Martin, Rich Thompson, Charlie Wiecha
05/Mar/2002	1.1	Updated with alternative flows	Charlie Wiecha
13/Mar/2002	1.2	Categorized and refined sub flows	Ravi Konuru and Eilon Reshef
13/Mar/2002		Added comments	John Lucassen

Table of Contents

Use Case Report: Customized Producer

1. Definition of the Customized Producer use case

In this use case, the Producer's definition includes specific information (such as properties, operations, and/or adaptation points) for enabling programmatic access by the Consumer to the Producer or its inputs and outputs for customization, rather than end-user or admin customization through the visual service itself (e.g. edit page in portlet). Depending on the capabilities of the Producer, its definition may describe one or more of the structure of (1) its input/output data, (2) properties controlling the output (preferences for interaction techniques or formatting (e.g. calendar mode), (3) the properties in the input or the output stream that can be manipulated by the consumer and (4) the operations it needs to call to perform various actions such as filtering queries and validating data.

The Consumer may wish to adapt, or customize the settings of any of these aspects of the Producer's interface and/or of the output it returns.

1.1 Brief Description

The Customized Producer use case captures the scenarios in which the Consumer is able to initialize Producer components with application specific context (such as interaction preferences, output sub-setting preferences…), under programmatic control, and retrieve application specific results from the Producer components upon completion of some unit of their work.

The Customized Producer use case extends the Embedded use case by providing for application specific context, and by allowing for that context to be both passed into the component by the Consumer, and returned from the Producer to the Consumer during the Producer's execution.

Note that in this use case, the Consumer does not republish itself as a web service -- this step is left to the Republished use case. Thus the consumer executes as a potentially platform-specific application such as a portal server or a J2EE or .Net application.

2. Actors

There are three actors in this use case:

- Producer: one or more WSIA web services
- Consumer: a piece of code or an application that can communicate with WSIA services, instantiates and controls interaction with the Producers on behalf of End-Users
- End-User: a person who interacts directly with the output of the Consumer e.g via a browser.

3. Flow of Events

Two flows are considered in this use case:

- Static definition of a Producer's context by the Consumer through use of an appropriate Producer-specific persistent store, and
- Dynamic definition of a Producer's context, computed by the Consumer using relevant data about the End-User's profile, interaction state, or other information.

3.1 Basic Flow

- The End-User enters a URL pointing to the Consumer into a browser
- The Consumer in Figure 4.1 is instantiated by an application server, which creates a session on it for interaction with the End-User
- The Consumer instantiates WSIA proxies as shown in Figure 4.1 for each of the Producer services it wishes to include in the page to be returned to the End-User, and creates a session on each to represent interaction with the End-User. Note that instantiation of a proxy may not be required if the Consumer chooses to interact with Producers via a framework such as WSIF -- the Web Services Invocation Framework-- which allows direct access to services without use of proxies.
  - The Consumer obtains the definition of the context information supported by each Producer. This information may be obtained statically from a directory service such as UDDI for the given Producer or dynamically by querying the Producer service itself. *Separate cases for various forms of customizing data, presentation preferences, and generated output are given below in the subflows sections.*
- The Consumer may initialize the Producer with initial data needed for its operation.
- The Consumer requests output from the Producer service, passing it the appropriate context information.
- The Producer responds with its output, along with any context information changes that may occur as a side effect of the assignments made by the Consumer.
- The returned page is adapted in two aspects:
  - Embedded URLs and form actions are rewritten to refer back to the Consumer and allow it to redirect them to the appropriate Producer. (This can be done either by the Producer or by the Consumer, as described in the subflows below).
  - The output may be customized by embedding additional output, removing optional output, replacing content such as optional images, resizing images, altering fonts and colors, and so on. (This can be done either by the Producer or by the Consumer, as described in the subflows below).
    - During the customization process, additional communication between the Consumer and the Producer (most likely, to exchange data) may occur in parallel to the communication described in this use case.
- The Consumer assembles the page resulting from output from the Producer, along with additional page content originating from other Producers or from the Consumer itself.

- The consumer may create an integrated form from multiple providers. In this case, the adaptation information should have sufficient information remove redundant submit buttons either at the producer or the consumer. The consumer may need to perform disambiguation of form fields to create an integrated page. Further, an interaction such as submit must be parsed at the consumer to send the right fields to the right provider.
- End-User interactions with the Consumer's page are directed back to the Consumer. At the Consumer, rewritten URLs are decoded to delegate the End-User action back to the appropriate Producer component. The Consumer invokes the Producer component passing any arguments provided by the page's action invocation as opaque data to the Producer. The consumer may look into, modify, or add to the arguments being passed to producer to dynamically customize the request to one or more provider services.
- The Producer, on returning from the action invocation, may pass updated context information back to the Consumer indicating the effect of the user action on the published context maintained by the Producer.
- If the Producer completes its interaction with the End-User, the Consumer may obtain any “output” data from the Producer.

3.2 Sub Flows

3.2.1 General

Many of the sub-flows below are presented according to two parallel “paradigms”: Operation/Property adaptation and Stream-based adaptation. In the former case, the Producer provides meta-information that describes operations and data (a.k.a., Properties) that affect its behavior. The Consumer adapts the behavior of the Producer by sending and receiving appropriate data via operations and properties. In the latter case, the Producer supplies meta-information about the structure of its stream (both output and input from end-user). This information is then used by the Consumer to manipulate the stream.

3.2.2 Static definition of Producer context by the Consumer

In this alternative flow, the Consumer proceeds as above through instantiation of the Producer and creation of a session for interaction with the individual End-User.

The context information to be passed to the Producer is determined statically in this flow by the Consumer through use of an appropriate Producer-specific persistent store.

[Multimedia sports portal] Maintain a user profile that tracks user interest and provides relevant information based on user profile.

[Traveler's checks] Customization can either be static (offline configuration, a typical portal scenario) or dynamic (a per-user or per-session configuration, a.k.a., personalization).

In this alternative sub-flow, the static context information determined above by the Consumer is overridden by End-User specific preferences, also obtained statically from a persistent End-User profile store. It is assumed that End-User preferences have precedence over those set on an application-specific basis by the Consumer for all users.

3.2.3 *Dynamic definition of Producer context by the Consumer*

In this alternative flow, the Consumer derives Producer-specific context information dynamically using available information on the End-User's preferences, on the current or historical interaction state, and any other information available to the Consumer.

[Financial charting] FAME Consumer application determines that stock plot is required for a given end-user, selects it from the UDDI directory, and sets iChart-specific preferences on it given parameters from the end-user's profile. No admin or end-user edit mode screens are required as these configuration parameters are exposed in the iChart service description directly. End-user interactions may override initial choices picked by the Consumer. Example Consumer specified preferences may include stock symbol, start/end dates, averaging options, etc. For calendar displays, Consumer may select options such as day/week/month view, US vs. European format (Sunday vs. Monday starts a week).

3.2.4 *URL rewriting*

3.2.4.1 Operation/Property Oriented adaptation:

To further customize the output, Producers generates appropriate output based on passed context information.

To enable redirection of all End-User actions back to Consumer’s page, Producer rewrites URLs in the page to link back to Consumer URL identified in the passed context information.

3.2.4.2 Stream Oriented adaptation

[Universal bank, Mortgage center, Product configurator] Consumer is able to make modifications to look and feel (skin) of the Producer to achieve its own branding goals such as a unified look and feel across Producers, *and to implement its marketing campaigns such as adding advertisements and cross-sell promotions that are personalized to the End-user and/or to the data entered by the user and/or returned by the Producer*.

3.2.5 *Prohibit, Permit and Constrain Look and feel adaptations*

Prohibit and Constraints are assumed to be only a hint to the consumer. The actual enforcement is outside the scope of the software stack for now.

3.2.5.1 Operation/Property Oriented adaptation

The Producer supplies a set of pre-defined properties – new values for these properties can be used by the Consumer to adapt the Producer’s behavior. The values for these properties may be checked by the Producer to ensure compatibility with business arrangement.

3.2.5.2 Stream oriented adaptation

[Beauty boutique, Insurance Enrollment, SmartBuyer, Multimedia sports portal] Consumer is precluded from making modifications to look and feel (skin) of the Producer to preserve branding requirements of the Producer.

3.2.6 Excluding elements

3.2.6.1 Operation oriented adaptation

The Producer supplies a set of pre-defined Boolean properties – new values for these properties can be used by the Consumer to exclude elements. The values for these properties may be checked by the Producer to ensure compatibility with business arrangement.

3.2.6.2 Stream oriented adaptation

The Producer supplies a set of locators (e.g., XPATH, character range) into each output stream. The consumer can use these locators to remove elements from the output.

3.2.7 Adding elements

3.2.7.1 Operation oriented adaptation

The Producer supplies a set of pre-defined “markup” properties – new values for these properties represents content that is provided to the Producer and incorporated into the pages. The values for these properties may be checked by the Producer to ensure compatibility with business arrangement.

[Mortgage center, Product configurator, Traveler's checks] Consumer requests elements to be added by the Producer either before generating output

3.2.7.2 Stream oriented adaptation

[Mortgage center, Product configurator, Traveler's checks] Consumer inserts elements into the output generated by the Producer as it passes through the Consumer.

The Producer supplies a set of locators (e.g., XPATH, character range) into each “gap” location within the output stream. The consumer can use these locators to insert elements into the output.

3.2.8 Replacing element values

[Product configurator] Consumer replaces logos or other branding elements in the Producer's display either before generating output using Operation oriented adaptations(3.2.7.1) or as the output passes through the Consumer using Stream oriented adaptations (3.2.7.2)

3.2.8.1 Operation Oriented adaptation

Before or at the time of requesting the output from the provider, the consumer first determines with particular elements in the provider's output that they wish to replace and with what content. To support this determination, the producer may have statically or dynamically provided

- a means of identifying a replaceable element (e.g. property name)
- the replacement constraints. (property constraints using e.g schema)

This information is used by the consumer in providing the right replacement content before the producer service produces the output.

3.2.8.2 Stream Oriented adaptation

The consumer first obtains information about the producer's output statically or dynamically.
This information includes

- A means of identifying a replaceable element in the stream (a locator e.g xpath)
- the replacement constraints. (property constraints using e.g schema)

This information is used by the consumer in replacing the replaceable element with the right replacement content before sending it to the end-user.

3.2.9 Editing elements

[Product configurator] Consumer desires to add an additional column to a table generated by the Producer based on the values of a key field, such as part number, and adding a column for corresponding values for a new field, such as price.

3.2.9.1 Operation oriented adaptation

Consumer passes pricelist back to the Producer and requests the output for the customized table.

3.2.9.2 Stream oriented adaptation

Consumer is given information by the Producer about the table structure in the producer's output stream to enable it to make the modifications locally.

3.2.10 Adding elements conditionally

[Mortgage center] Add new questions asked, depending on answers given to the original (Mortgage) questions -- in this use case on the same page.

3.2.10.1 Operation oriented adaptation

In this case, the consumer provides the new questions to be asked, their identities, their relative positions in the output, and the criteria under which they should be made visible.

3.2.10.2 Stream oriented adapation

3.2.11 Adding new options to existing questions

[**RK Combination of Adding elements and Adding values?**] . [Mortgage center] calculation for US locations, but Canadian lending institution. The business calculation has to be adapted too. Tweak how the calculation of the home price affects the final number (in scope?)

3.2.11.1 Operation oriented adaptation

3.2.11.2 Stream Oriented adaptation

3.2.12 Setting preferences for interaction technique selection and formatting

3.2.12.1 Operation oriented adaptation

Configure a calendar by selecting the preferred type and controlling display properties on the interaction technique such as day/week/month views or starting day of the week.

3.2.12.2 Stream oriented adaptation

This can be done at the consumer only all the views are available in the output and consumer can "make visible" the right view and the binding.

3.2.13 Reducing the need for user interaction [Same as 3.2.8 editing values?]

3.2.13.1 Operation oriented adaptation

[Mortgage center] Remove a question about property tax value and provide a Consumer-specified value. Tweak the generic property tax value to be the exact value for the given city.

3.2.13.2 Stream oriented adaptation

Consumer-computed values and filling in the stream

3.2.14 Redefining user action handlers

3.2.14.1 Operation oriented adaptations

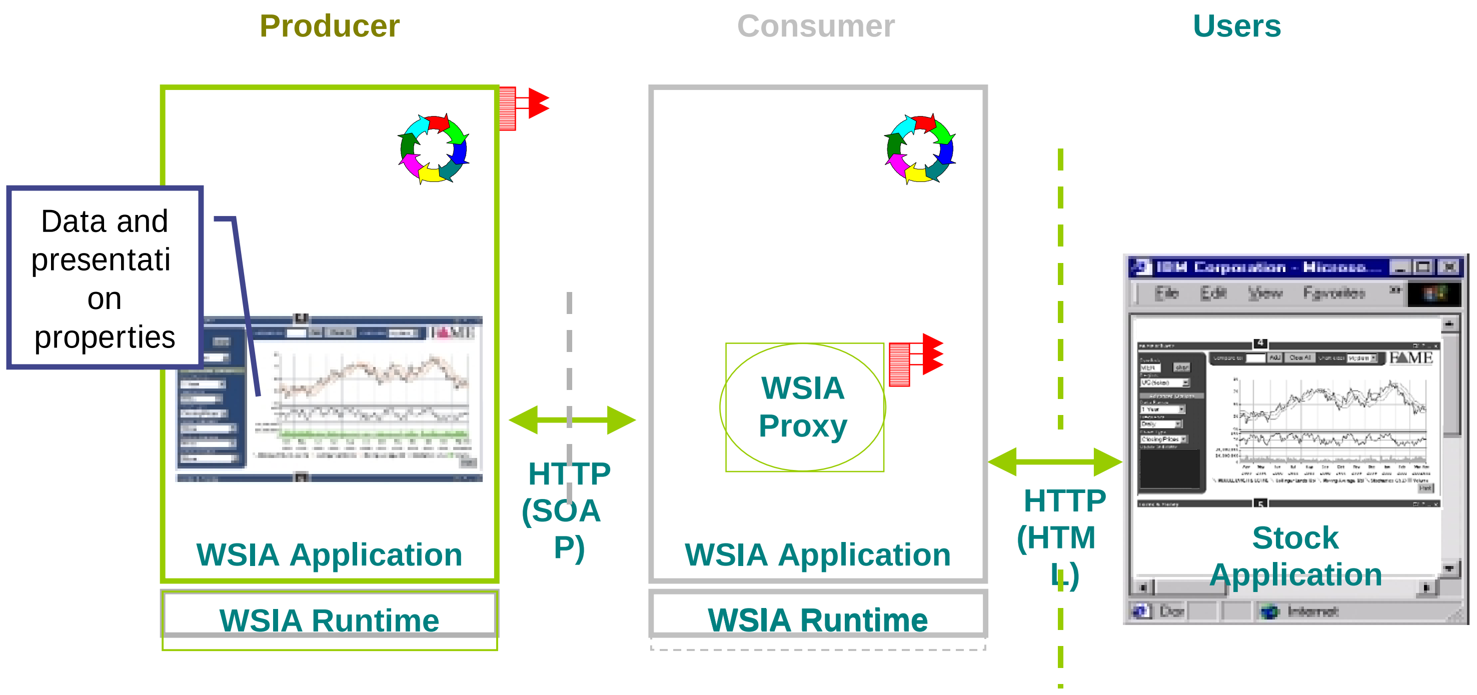
[Product configurator, Buy.com] Altering the function of the Add to Shopping Cart button, subflow 1: intercepted by the Consumer, subflow 2: caught by the Producer by action redirected back to the Consumer.

3.2.14.2 Stream oriented adaptations

Redefining user actions by interception at the Consumer

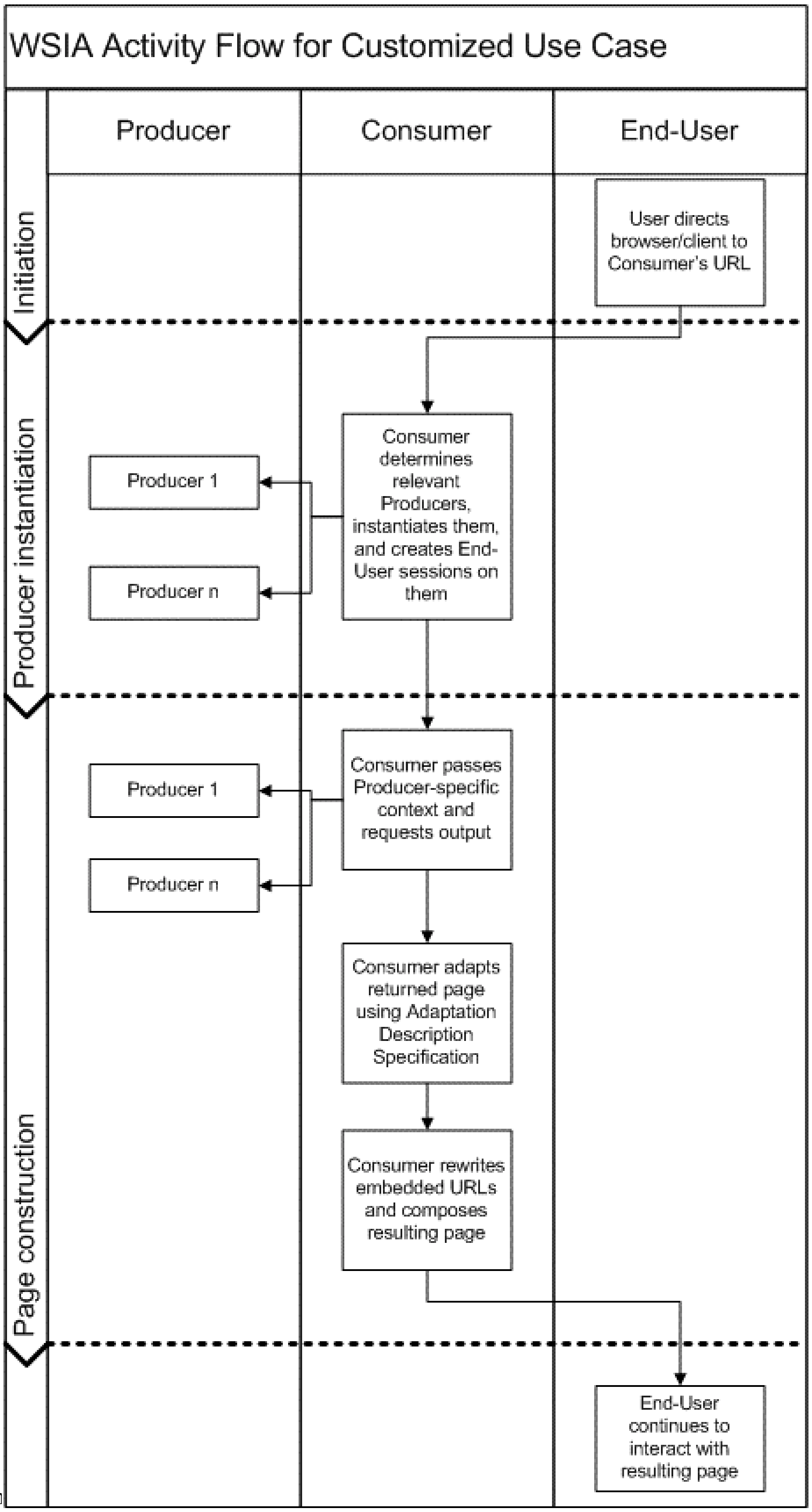
4. Diagrams

4.1 Relationship between Producers and Consumers in the Customized Use Case

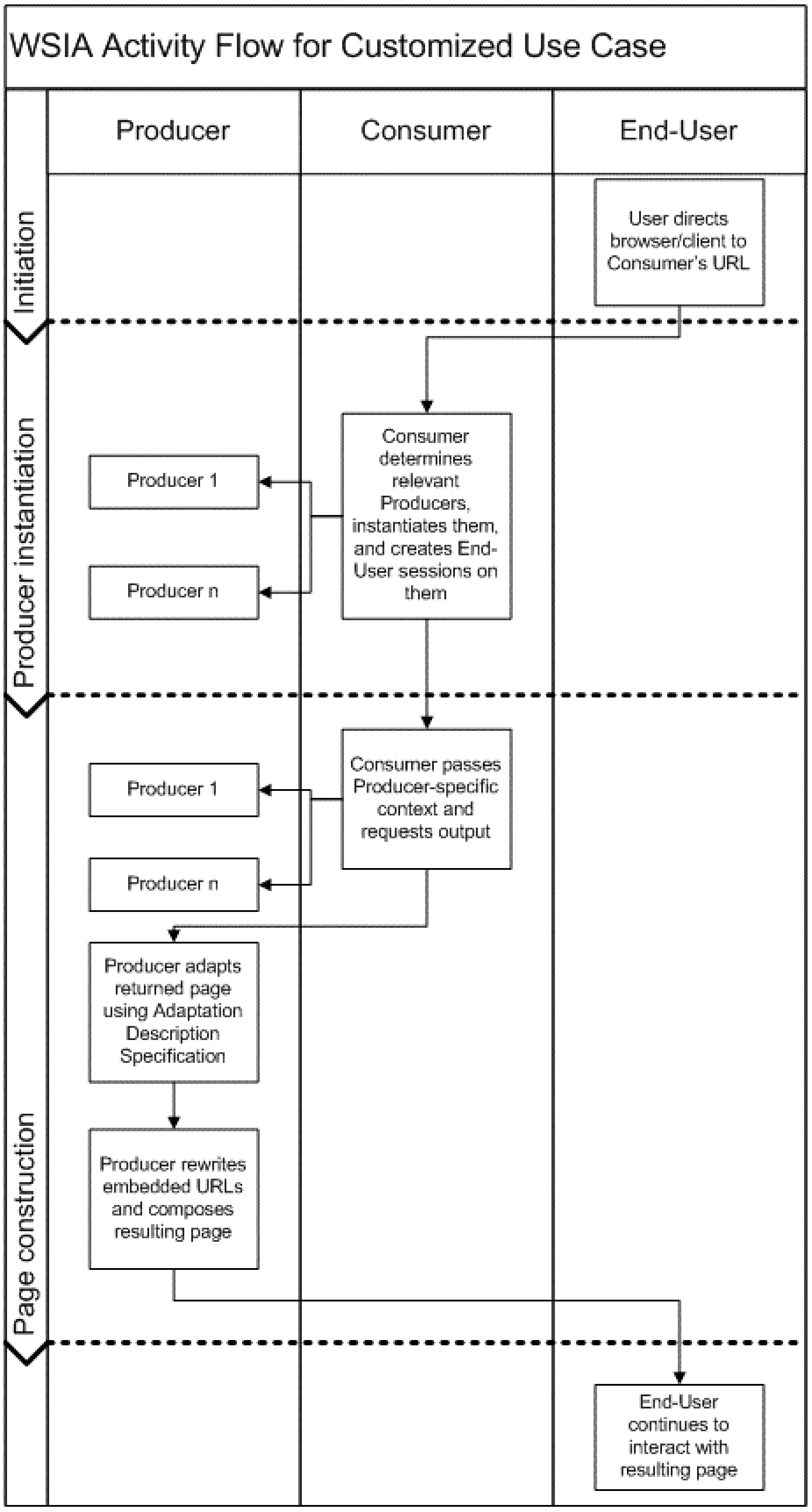


Producer
Consumer
Users
Data and presentation properties
WSIA Proxy
HTTP (SOAP)
HTTP (HTML)
WSIA Application
WSIA Application
Stock Application
WSIA Runtime
WSIA Runtime

4.2 Main flow for Customized use case


WSIA Activity Flow for Customized Use Case
Producer
Consumer
End-User
Initiation
Producer instantiation
Page construction
User directs browser/client to Consumer's URL
Consumer determines relevant Producers, instantiates them, and creates End-User sessions on them
Producer 1
Producer n
Consumer passes Producer-specific context and requests output
Producer 1
Producer n
Consumer adapts returned page using Adaptation Description Specification
Consumer rewrites embedded URLs and composes resulting page
End-User continues to interact with resulting page

4.3 Alternative flow diagram for Producer applied adaptation and URL rewriting


WSIA Activity Flow for Customized Use Case
Producer
Consumer
End-User
Initiation
Producer instantiation
Page construction
User directs browser/client to Consumer's URL
Consumer determines relevant Producers, instantiates them, and creates End-User sessions on them
Producer 1
Producer n
Consumer passes Producer-specific context and requests output
Producer 1
Producer n
Producer adapts returned page using Adaptation Description Specification
Producer rewrites embedded URLs and composes resulting page
End-User continues to interact with resulting page

4.4 < First special requirement >

5. PreConditions

[A precondition (of a use case) is a textual description of any constraints or dependencies that must be satisfied prior to entry of the use case.]

5.1 < Precondition One >

6. PostConditions

[A postcondition (of a use case) is a textual description of any constraints or dependencies that must be satisfied after termination of the use case.]

6.1 < Postcondition One >