Processes and Policy in Service-Oriented Architectures

David Bantz
Adjunct Professor
University of Southern Maine
Portland, Maine
David.Bantz@maine.edu
www.cs.usm.maine.edu/~bantz
February 2007
Objective

To familiarize you with process and policy representations and capabilities, as they may appear in Service-Oriented Architectures (SOA).

1. (opt) What is a Service-Oriented Architecture?
2. Why are computerized processes useful?
3. Why are computerized policies useful?
4. Summary and brainstorming
Who is this guy?

• Adjunct at Columbia Engineering 1974-1997
• Startup (CompuScan) 1970-1972
• IBM Research 1972-2005, 2006 - present (33+ years!)
  – office systems, computer graphics, printing, wireless LAN,
    network and autonomic computing and client management
  – 27 issued patents, 28 more in the pipeline
• Adjunct Professor at USM. Teaches Java, algorithms, project
  management and Web Services
• Likes woodworking, programming and (sometimes) taking care of
  old houses.
A Brief History of Enterprise IT

Many enterprise functions (manufacturing, management, financial) have millennia-long history, having evolved to effective and efficient structures to support the enterprise. IT has a short history (decades) with a lot of technology-driven churn.

IT first supplemented or replaced manual processes with batch programs. These programs were isolated from one another (“silos”) and thus were relatively easy to maintain. Then came the need for end-to-end processing – “straight-through processing (STP).” This coupled previously-isolated programs and mired the enterprise in integration issues.
A Brief History of Enterprise IT (contd.)

In response to STP, some enterprises standardized on a common platform for all programs, easing the integration issue but causing enormous amounts of rewrite and, in some cases, loss of precious business logic.

When Web Services (a new technology) was ready for prime time, its first application was as an integration technology that did not require a common platform.

“Service-Oriented Architecture” is a pattern, using Web Services, that is focused on integration and reuse. It exemplifies a mindset about Enterprise IT – Componentize, Glue, and Script (CGS) to implement business processes.
Business needs drive IT

• For some time now, the basic technology provided by hardware, system software, programming languages and middleware has been sufficient to meet the needs of business.

• But there continues to be a gap between business abstractions and the implementation detail that is the concern of software developers.

• This gap can be (partially) closed through the introduction of executable business processes and policies. From the perspective of IT, they are like scripting; from the perspective of the business, they are formalizations of business design.
Workflow

- An old concept – essentially a data flow representation. This had been used in manual system analysis.
- Represents processing steps and data items (e.g., records). A data flow diagram shows how data is routed from one step to another.
- Can represent alternative steps, parallelism, correlation and “compensation” (undo).
- Can represent processing steps performed by people, and glue between steps.
- Can be described in an XML-based language – WS-BPEL – which can be interpreted by a “workflow engine.”
Scripting – Business Process Choreography
WebSphere Business Modeler → WS-BPEL
BPEL skeleton (Graham)

<process name="purchaseOrderProcess">
  <partnerLinks>
    <partnerLink name="buyer"
      partnerLinkType="plt:purchaseOrderPartnerLinkType"
      myRole="seller"/>
  </partnerLinks>

  <partners>
    <partner name="skatestownCustomer">
      <partnerLink name="buyer"/>
    </partner>
  </partners>

  <variables>
    <variable name="poSubmissionRequest"
      messageType="pos:poSubmissionRequest"/>
  </variables>

  <correlationSets>
    <faultHandlers>
      <eventHandlers>
    </process>
BPEL activity (Graham)

<sequence>
  <receive partnerLink="buyer" portType="pos:poSubmissionPortType"
    operation="poSubmission createInstance="yes"
    variable="poSubmissionRequest">
  </receive>
  <flow>
    <links>
      <invoke partnerLink="local" portType="skt:skatestownPortType"
        operation="validatePurchaseOrder"
        inputVariable="poSubmissionRequest"
        outputVariable="validatePurchaseOrderResponse">
        <source linkName="validPurchaseOrder"
        <source linkName="invalidPurchaseOrder"
      </invoke>
    </links>
    <sequence name="replenishStock">
      <assign>
        <copy>
          <from variable="poSubmissionRequest" part="purchaseOrder"/>
          <to variable="orderSuppliesRequest" part="orderSupplies"/>
        </copy>
      </assign>
      <while condition="bpws:getVariableData('checkLocalStockResponse',
        'available')=false">
        <sequence>
        </sequence>
      </while>
    </sequence>
  </flow>
</sequence>
When Do You Use a (BPEL) Process?

• When the process definition captures a business design. For example, when you want to communicate it to a business designer.

• When you think it might change, say as a consequence of regulatory action, mergers and acquisitions or as a result of a new business model.

• When exception cases or asynchronous events do not dominate.
What is “Policy?”

• The term “policy” refers to IF-THEN rules that are stored in one place for an entire organization. Any application can consult these rules for “guidance.”

• “Guidance” consists of data that the application will use to alter its behavior.

• Example: Should I throttle down my server?
  Guidance: yes, if the number of simultaneous users is greater than 100 and it is prime time

• Policies can concern IT-related behavior (e.g., how much resource can I consume?) too.
Why store only one copy of a policy?

- A single copy makes it possible to change the behavior of all of IT with just one policy change.
- If all policies are stored together, and in a formal representation, this makes it possible to check them for completeness and consistency.
- If all policies are stored together, and you need to change many of them at once (say, because of an acquisition) you can deploy the new set of policies easily.

If policies are implemented as code in applications, a change in policy implies a horrendous, error-prone maintenance task.
PMAC with Unsolicited Guidance

Policy Store → Policy Evaluation

“sensors” → “effectors”

Properties

“Autonomic Manager”

Application
SPL syntax

Policy {
    Header {
        Name = String
        Version = String
        DecisionName = String
    }
    BusinessValue = int
    Scope { of applicability }
    Description { textual description }
    NlsDescription { resource bundles }
    Condition { IF clause }
    Decision { THEN clause }
    TimePeriod { active time period }
}
SPL example
(the ACPL is horrible)

Policy {
  Header {
    Name = "ThrottleDown"
    DecisionName = "D"
  }
  Scope { Name = "N" }
  Condition { NUsers > 100 }
  Decision { THEN true }
  TimePeriod {
    TimeOfDayStart = 07:30
    TimeOfDayEnd = 17:30
    DaysOfWeek = 2,3,4,5,6
  }
}
When Do You Use a Policy?

• When a process must be responsive to a variable business condition. For example, during tax season.
• When the condition can be a moderately complex function of measurable business variables and time.
• When you are trying to capture a customer- or enterprise-specific constraint, and you want your process to be generic across customers and enterprises.
• When the enterprise refers to the condition as a “policy.”
Autopilot: a meta-manager
Summary

- Process and policy definitions help bridge the gap between IT and business design.
- Externalizing process definitions lets IT help business be more flexible.
- Externalizing policy definitions helps process definitions be more generic.
- WS-BPEL is fairly mature (2.0 approved 1/31/07!) but policy standards are still evolving.
Sources

- Building Web Services with Java, 2\textsuperscript{nd} ed., Graham et. al., Sams Publishing 2005
- WS-BPEL: www.oasis-open.org
- IBM WebSphere Business Modeler: www.ibm.com/software/integration/wbimodeler
- PMAC: www.alphaworks.ibm.com/tech/pmac
Backup
CGS is a pattern for IT

- CGS promises high-speed implementation of new business processes, provided that the components already exist.
- CGS doesn’t use IT resources (processing, memory, storage and communication) efficiently, but that’s OK because these resources are much less expensive than manpower.
- CGS establishes the need for new technologies and products
  - Scripting languages for glue (very hot topic lately)
  - Backbone products like the Enterprise Service Bus
  - Workflow design and implementation
The Enterprise Service Bus
(courtesy IBM)
Service-Oriented Architecture (SOA)

• SOA is CGS with specific choices: loosely-coupled Web Services components, SOAP as the message format and workflow engines for business process implementation.

• An SOA-based business process is assembled, rather than coded. It is both an overlay on the IT infrastructure of an organization and an infrastructure in itself.

• SOA makes integration easy for IT staff and business process designers. IT can now respond to your needs for dynamic reorganization, acquisition and fast reaction to customers’ needs.

• There is still some magic in getting the components right, but this can be an evolutionary process.
Componentize, Glue and Script (CGS)

- **Componentize**: Have your developers and software vendors design and build reusable functions. Don’t constrain their implementation by platform or language. Don’t force all of the interfaces to match exactly, but do use XML.

- **Glue**: Transform XML messages between interfaces of functions with little scripts or data-transformation languages (e.g., XPath). You can even augment a message through database access.

- **Script**: Have your business process designers write a script to string functions together into the workflow needed to build a business process.