Web Services Choreography and Process Algebra

29th April 2004

Steve Ross-Talbot

Chief Scientist, Enigmatec Corporation Ltd

Chair W3C Web Services Activity

Co-chair W3C Web Services Choreography

Agenda

- Orchestration vs Choreography
- WS-BPEL
- WS-CDL
- Underpinnings
- Status
- Q&A

Orchestration vs Choreography

- Consider a dance with more than one dancer.
- Each dancer has a set of steps that they will perform. They orchestrate their own steps because they are in complete control of their domain (their body).
- A choreographer ensures that the steps all of the dancers make is according to some overall scheme. We call this a choreography
- The dancers have a single view point of the dance.
- The choreography has a multi-party or global view point of the dance.

Orchestration vs Choreography

- Orchestration is about describing and executing a single view point model.
- Choreography is about describing and guiding a global model.
- You can derive the single view point model from the global model by projecting based on participant.

WS-BPEL and WS-CDL

- WS-BPEL
  - Orchestration implies a centralized control mechanism.
- WS-CDL
  - Choreography has no centralized control. Instead control is shared between domains.

Orchestration of Web Services

- **The Oasis WS-BPEL TC**
- Summary: Orchestration of web services and recursive composition thereof.
- Style: Scoped programming language (BPEL) with behavioural interfaces (Abstract BPEL).
- Uses: Orchestration of Web Services in a single domain of control (i.e. order flow within institution).
- Status: Currently X issues to resolve and based on WSDL1.1 and some proprietary specs. Due to deliver Q4.
- Issues: Licensing. Based on some proprietary specifications

WS-BPEL

- Is a Web Service
  - Runtime semantics
  - Centralised orchestration
- Abstract
  - Defines end-point protocols
- Executable
  - Executes the necessary WSDL calls effecting message exchange between services
- Benefits
  - Higher reuse of WSDL collateral

WS-BPEL

- Sequence,
- Fork,
- Join,
- Parallel threads,
- Computation (Turing Complete)

WS-BPEL - Problems

- Centralised execution
- Lack of formal semantics
- Non-scalable (requires the concept of dual connectivity)
- Non-collaborative

Choreographing Web Services

- **W3C Web Services Choreography Working Group**
- Summary: Describing peer to peer interaction in a global model by means of a CDL
- Style: Formalized description of external observable behavior across domains
- Use for: Modeling cross domain protocols, protocol enforcement, skeletal code generation (i.e. for FIX)
- Status: Requirements document formally published, Model Overview document published to mailing list. Due to deliver end 2004.

What is a Choreography

- WS-Choreography concerns the collaboration protocols of cooperating Web Service participants
  - WS act as peers
  - WS interact in long-lived, stateful & coordinated fashion
- A WS-Choreography description is a multi-participant contract that describes, from a Global Viewpoint, the *common* observable behavior of the collaborating WS participants
- WS-CDL is a language in which such a contract is specified
  - Standardization underway in the W3C Choreography WG

Using a WS-CDL

- promote a common understanding between WS participants;
- automatically guarantee conformance;
- ensure interoperability;
- increase robustness;
- generate code skeletons.

Benefits of a WS-CDL

- more robust Web Services to be constructed;
- enable more effective interoperability of Web Services through behavioral multi-party contracts, which are choreography descriptions;
- reduce the cost of implementing Web Services by ensuring conformance to expected behaviour;
- increase the utility of Web Services as they will be able to be shown to meet contractual behavior.

Overview of WS-CDL

- Interactions
- Channels
- Participants
- Roles
- State

WS-CDL Approach

- Simple contract-like mechanisms are exhibited in the literature for capturing
  - Deadlock-freedom (Kobayashi, 99, 00)
  - Liveness (Kobayashi, 01; Yoshida, et al, 02)
  - Security (Abadi et al; Cardelli and Gordon; Berger, Honda, Yoshida)
  - Resource management (Tofte; Kobayashi; Gordon and Dal Zillio; Yoshida, et al)
- A contract language that guaranteed even basic versions of these properties (at the compatibility level) then that would be a significant advance over the state of the art.

WS-CDL Approach

This work needs to be carried out using formal basis. To the extent possible, *technical* design deliberations can and should be a matter of calculation.

Mobile process calculi provide a natural candidate.

Web service Implementation	Process
Does roughly what client wants it to do	Bisimulation 'approximation'
Contract	Behaviorial type

Why process calculi?

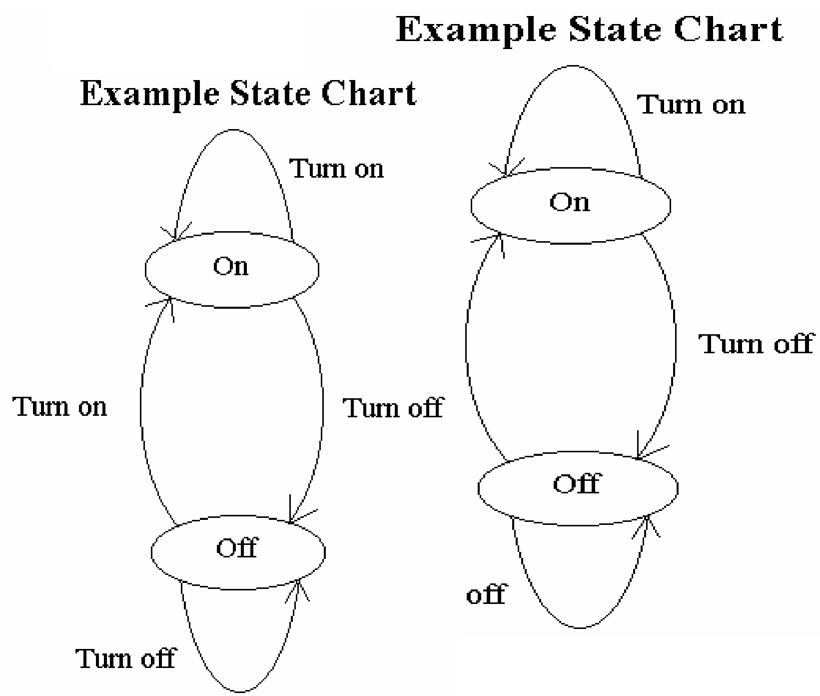
Model	Completeness	Compositionality	Parallelism	Resources
Turing Machines	☑	☒	☒	☑
Lambda	☑	☑	☒	☒
Petri Nets	☑	☒	☑	☑
CCS	☑	☑	☑	☒
π	☑	☑	☑	☑

Global Models

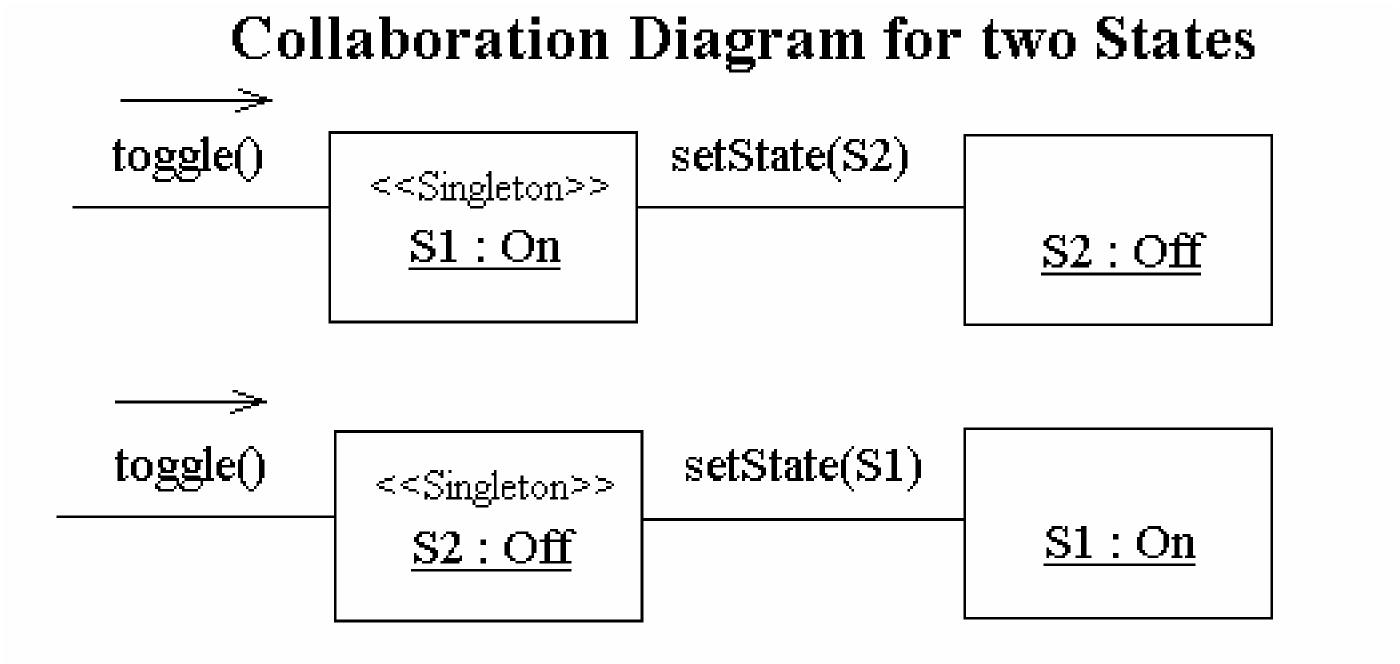
STATE TRANSITION TABLE FOR A GARAGE DOOR OPENER

Current State	INPUT	Effect	Next State
Door closed / Motor off	Button Pressed	Start Motor	Motor Running Up
Motor Running Up	Door Open Detected	Stop Motor	Door Open Motor Off
Motor Running Up	Button Pressed	Stop Motor	Door Partially Open/ Motor Off
Door Partially Open/ Motor Off	Button Pressed	Start Motor	Motor Running Down
Door Open / Motor Off	Button Pressed	Start Motor	Motor Running Down
Motor Running Down	Door Closed Detected	Stop Motor	Door Closed Motor off
Motor Running Down	Button Pressed	Stop Motor	Door Partially Closed / Motor Off
Door Partially Closed / Motor Off	Button Pressed	Start Motor	Motor Running Up

STAT
Current St
Door clos Motor o
Motor Run Up
Motor Run Up
Door Parti Open/ Motor
Door Ope Motor O
Motor Runni Down | Detected | Motor | Motor off
Motor Running Down | Button Pressed | Stop Motor | Door Partially Closed / Motor Off
Door Partially Closed / Motor Off | Button Pressed | Start Motor | Motor Running Up



Global Models



Global Models

Example flow for Pre-Trade Allocation (using Allocation Instruction message)

Initiator			Respondent
	→	New Order-Single (OrderQty=35000)	
	←	Execution Report (ExecType = "0" [New]	
	→	Allocation Instruction (AllocType=" Prelimina provided without MiscFees or NetMoney)	
	←	Allocation Instruction Ack (AllocStatus=R Processed)	
	←	Allocation Instruction Ack (AllocStatus=Accep	
	←	Execution Report (ExecType = "F") [Trade] (optional Execution Report (ExecType = "3")	

Good Till Order – Warehouse Until Filled Using Pre-Trade Booking Instruction

Day 1 – entire part-filled quantity is warehoused

BuySide				SellSide	
	→	New order single *GTBookingInst = 1*	→		*1. Buyside sends new GT order with instruction to warehouse any part-filled quantity until the order fills or expires (i.e. GTBookingInst is 1).*
	←	Execution reports *(new…partial fills)*	←		*2. Sellside accepts the order, then sends 1 or more partial fill execution reports.*
	←	Execution report *(done for day)*	←		*3. Sellside sends a "done-for-the-day" (DFD) execution report when execution completes for the day.*
	←	Allocation report *(AllocReportType = 5)*	←		*4. Sellside sends a warehouse recap allocation report.*
					Note a 'warehouse instructon' allocation instruction message from the buyside is not required at this point due to the use of GTBookingInst when placing the order

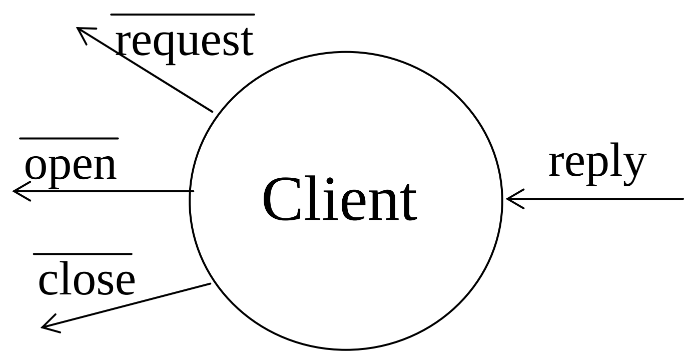
Day 2 – further executions; entire part-filled quantity is again warehoused

BuySide				SellSide	
	←	Execution reports *(new…partial fills)*	←		*2. Sellside sends 1 or more partial fill execution reports.*
	←	Execution report *(done for day)*	←		*3. Sellside sends a "done-for-the-day" (DFD) execution report when execution completes for the day.*
	←	Allocation report *(AllocReportType = 5)*	←		*4. Sellside sends a warehouse recap allocation report.*
					Note a 'warehouse instructon' allocation instruction message from the buyside is not required at this point due to the use of GTBookingInst when placing the order

WS-CDL Global Models

- A sequential process

$$Client(open,close,request,reply) = \overline{open}.\overline{request}_1.reply_1.\overline{request}_2.reply_2.\overline{close}.0$$

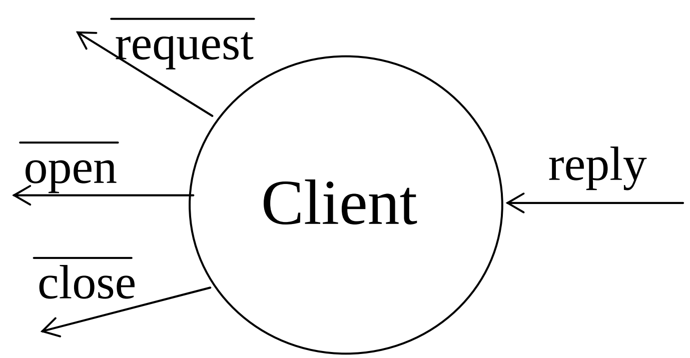


WS-CDL Global Models

- A repetitive process

$$Client(open,close,request,reply) =$$

$$\overline{open}.\overline{request}_1.reply_1.\overline{request}_2.reply_2.\overline{close}.Client(open,close,request,reply)$$

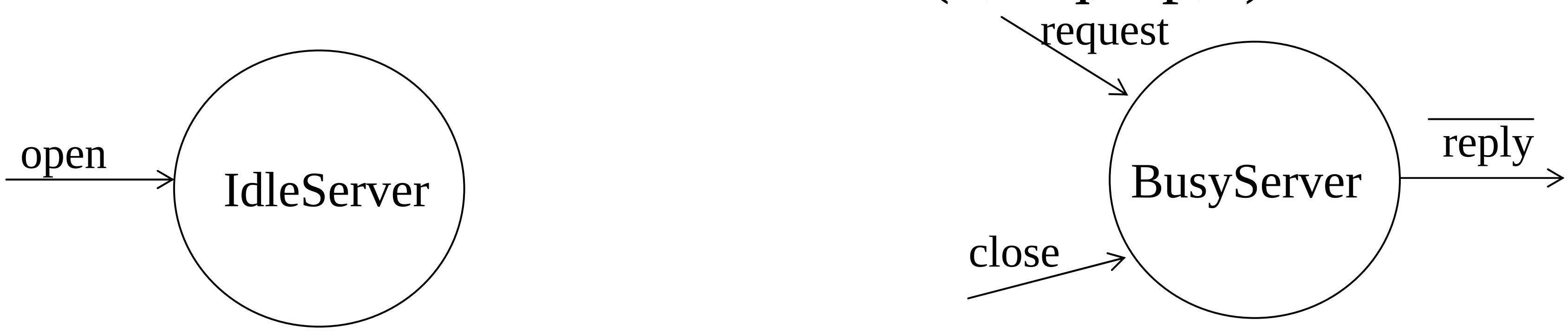


WS-CDL Global Models

- A process with choices to make

IdleServer(o,req,rep,c) =
o.BusyServer(o,req,rep,close)

BusyServer(o,req,rep,c) =
req.rep.BusyServer(o,req,$\overline{rep}$,c) +
c.IdleServer(o,req,rep,c)



WS-CDL Global Model

- Communication, Concurrency and Replication

SYSTEM = (!Client | IdleServer)

When $Client_i$ has started an exchange with IdleServer

$Client_i$ | IdleServer

$Client_i$ | BusyServer

No other Client can then communicate with the server

$Client_j$ | IdleServer

$Client_j$ | BusyServer

…..

Until $Client_i$ has finished and the server is once again Idle

WS-CDL and the pi-calculus

Operation	Notation	Meaning
Prefix	**π.P**	**Sequence**
Action	**a(y), $\overline{\text{a}}$(y)**	**Communicati**
Summation	**a(y).P + b(x).Q** **$\sum \pi_{i(}P_{i}$**	**Choice**
Recursion	**P={…..}.P**	**Repetition**
Replication	**!P**	**Repetition**
Composition	**P | Q**	**Concurrency**
Restriction	**(νx)P**	**Encapsulation**

Collapse send and receive into an interact on channels

WS-CDL and the pi-calculus

- Static checking for livelock, deadlock and leaks
  - Session types and causality
- Robust behavioral type system
  - Session types

WS-CDL - Status

- Where are we today?
- Working Draft V2
- Looking for comments
- Lots of work with vertical standards
- Looking to last call end Q404

WS-CDL Summary

- Global model
  - Ensured conformance
- Description language
  - Not executable
- Tools
  - Generators for end points
  - Advanced typing
- Status
  - Moving for last call end of Q404

References

- WS-CDL Working Draft
- WS-CDL Overview
- BPEL4WS 1.1
- Enigmatec