Provisioning Object-oriented Service Clouds for Exertion-oriented Programming

Michael Sobolewski
SORCERsoft.org, AFRL/WPAFB

May 9, 2011
Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and FSOOA
- SORCER metaprogramming and programming — EOL, VOL, VML
- SORCER Operating System (SOS) and FMI
- SORCER Virtual Processor and Provisioning
- Conclusions

Whatever we may want to say, we probably won't say exactly that!

Marvin Minsky
From AI to Metacomputing

• AI/Expertalk (PAN/PW/UŚ) – 1971-1989
• DICETalk (CERC/WVU, DICE/DARPA) – 1989-1994
• CAMnet (GE GRC/DARPA) – 1994-1995
• Agile Castings (GE GRC/DARPA) – 1995-1998
• WCE: UNS Notebook (GE GRC/AE), GE Plastics Calculator, EMPIS (GE PS), ++, – 1997-2000
• FIPER (GE GRC/NIST) – 1999-2003
• SORCER (TTU – 2002-2009, SORCERsoft/AFRL/++) – 2007-...
Computing Science

Computing's core challenge is how not to make a mess of it.

Edsger Dijkstra
Process Expression

- *Computer science is the science of process expression*  
  Karl Fant

- Process expression
  - Symbolic expression (Language)
    - grammar or metamodel (UML Behavior Diagrams)
  - Physical expression (Actualization)
    - computing platform
      - Programming environment
      - Operating system (Command, OO, SO)
      - Processor (native or virtual)
Process Expression

- Persian abacus (600 BC)
- Algorism (alKhowarizmi, 825) (algorists vs. abacists)
- Mathematics (Hilbert, 1920) (expressions complete, consistent, decidable)
- Algorithm – flowchart (Markov, 1954)
- Case-based, rule-based, and connectionist expressions
- Logic circuits (programmable FPGA, FPAA)
- Object interaction (object orientation)
- MOF/UML (M2, behavior diagrams)
- Service federation (service-orientation—exertions)

*You don't understand anything until you learn it more than one way.*

Marvin Minsky
Agenda

• Intro: computing science & process expression
• Distribution, object & service orientation
• Transdisciplinary computing processes — SO Platform
• C/S, SOA, SPOA, SOOA and FSOOA
• SORCER metaprogramming and programming
  — EOL, VOL, VML
• SORCER Operating System (SOS) and FMI
• SORCER Virtual Processor and Provisioning
• Conclusions
Terminology & Technology

• “The computer is the network.” vs. “The network is the computer.” (eight fallacies of network computing)
  – MS/IBM: The network is the App server.
  – Oracle: The network is the database.

• “A distributed OO system” vs. “An OO distributed system”
  – A distributed OO system – implicit network (network transparency)
  – An OO distributed system – explicit network
Composition Granularity

Abstractions of Programming Components

Mike Sobolewski
Agenda

• Intro: computing science & process expression
• Distribution, object & service orientation
• Transdisciplinary computing processes — SO Platform
• C/S, SOA, SPOA, SOOA and FSOOA
• SORCER metaprogramming and programming
  – EOL, VOL, VML
• SORCER Operating System (SOS) and FMI
• SORCER Virtual Processor and Provisioning
• Conclusions
Transdisciplinary (TD CE) Process
Leveraging resources and reuse for R&D growth

Ops: apps, tools, utilities -> **programs**
Metaprogram -> program of **programs**
(process expression by other process expressions)
By providing breakthrough product design technology, FIPER will significantly reduce product creation costs and time to market by 20-50%, while improving design robustness. (NIST $21.5 million)
FIPER Metaprogramming Domain

System Design

Subsystem Design

Component Design
Service-oriented Platform

Reducing complexity by higher level abstractions
Virtualization Dependencies

Service Providers – V1

Service Containers – V2

Object Platforms – V3

Command Platforms – V4

Native Platforms

Service Processor (Service Cloud)

Service-oriented OS

EO Programming (EOL)
Var Modeling (VML)
VO Programming (VOL)

EOL - service collaborations
VOL- multifidelity evaluation compositions
VML - multidisciplinary var-oriented composition
SORCER SO Platform

Service requestors (exertions) – commands of the SO processor (C0-C5)
SO program – an exertion executed by the SOS shell
SV – service virtualization, PV – platform virtualization

Service Requestors (Var-Models, Vars, Exertions)
Service Operating System
SOS Service Providers:
- Tasker, Jobber, Spacer, Provisioner, ODProvisoner, Cybermode, Gridier, Caller, Methoder,
- FileStorer, MetadataStorer, ByteStorer, Replicator, Sweeper,
- Tracker, Splitter, LocoManager,
- Authenticator, Authorizer, Policier,
- WsRpcRelayer, WsDocRelayer, JxtaRelayer, Surrogator, Cataloger,
- Logger, Notifier, Monitor, Persister

Service Containers

Service Providers:
- Evaluators, Filters, Differentiators, DS Service Beans

Command Native Platforms

Object Virtual Platforms

Command Virtual Platforms

Command Native Platforms
Agenda

• Intro: computing science & process expression
• Distribution, object & service orientation
• Transdisciplinary computing processes — SO Platform
• C/S, SOA, SPOA, SOOA and FSOOA
• SORCER metaprogramming and programming
  — EOL, VOL, VML
• SORCER Operating System (SOS) and FMI
• SORCER Virtual Processor and Provisioning
• Conclusions
Quantum Jumps in Platform Complexity

Sequential Programming
+ order
runtime: batch processing, OS

Multi-threaded programming
- order
+ parallelism
runtime: + concurrency support

Multi-process Programming (time-sharing)
- context
+ SW isolation (safety)
runtime: + interprocess communication (pipes, sockets)

Multi-machine Programming (client/server) (DICE, CAMnet, Agile Castings)
- global state, security, trust
+ HW isolation, scalability
runtime: + secure interprocess communication (RPC), trusted mobile code (proxying), virtual file system, disconnected operations, leases, transactions, distributed events, deployment control

Grid Programming (SOA) (FIPER)
- resource setup
+ resource utilization, collocation, distributed resource sharing
runtime: +batch processing (job schedulers) using individual OSs to aggregate CPUs for conventional programs execution

Metaprogramming (FSOOA) (SORCER)
- SO federated programming, DI deployment setup, untrusted mobile code, class loading
+ SO, service federation spontaneity, behavioral transfer, autonomic provisioning
runtime: + service orientation: SO processor, SO OS, and SO programming model
Client/Server
SOA

Service Registry

Requestor

Register

Lookup

Proxy

Def

Service

Service Provider
SOOA Terminology

- **Service type** – interface type (service)
- **Service object** – an object implementing its remote service types (services) accessible via its proxy object
- **Service provider** – service object accepting *remote invocations* on one or more its service types
- **Service bean** – local object (POJO) implementing interface types
- **Service container** (service node or cybernode) – service object that *deploys and manages* one or more service providers
- **Discovery** – finding out a service registry
- **Lookup** – finding out a service proxy object
SOOA Three Neutralities & BT in SOOA

- Service Registry
  - Discovery & Lookup
  - Discovery & Register
- Service Requestor
  - Exported Classes
- Service Provider
  - Code Server
  - Mobile Proxy Object
Neutralities

• Implementation
  – Service type (not IDL description)

• Location
  – Dynamic (not static, no endpoints)

• Wire protocol
  – Any (not fixed, e.g. SOAP)

• Data format
  – Generic (*Context* interface) with conversion on external boundaries to XML (no XML within SORCER)
Read-write vs. Remote Invocation
Six RPC Generations

• First generation RPCs: Sun RPC (ONC RPC) , DCE RPC
  – language, architecture, OS independent
  – IDL
• Second generation RPCs: CORBA, Microsoft DCOM-ORPC
  – adds object support
• Third generation: Java RMI
  – it is conceptually similar to the second generation but supports the semantics of object invocation in different address space
  – is built for Java only
  – fits cleanly into the language (interfaces, serialization)
  – no need for standardized data representation
  – with behavioral transfer
• Fourth generation RPCs: Jini Extensible Remote Invocation (Jini ERI)
  – dynamic proxies
  – dynamic configurations (dependency injection
  – security
• Fifth generation RPCs: Web Services RPC and the XML bandwagon
  – SOAP
  – WSDL
• Sixth generation RPCs: SORCER *Federated Method Invocation* (FMI)
  – invocation on multiple federating services (virtual metaprocessor)
Net-Centric FSOOA

Exertion Shell

Service Registry

Service Provider

Mobile Proxy Object

Exert

Exported Classes

Code Server

Deployed or provisioned

Exertion Shell

Exported Classes

Code Server

Service Provider
Agenda

• Intro: computing science & process expression
• Distribution, object & service orientation
• Transdisciplinary computing processes — SO Platform
• C/S, SOA, SPOA, SOOA and FSOOA
• SORCER metaprogramming and programming
  – EOL, VOL, VML
• SORCER Operating System (SOS) and FMI
• SORCER Virtual Processor and Provisioning
• Conclusions
Language – Mankind
Writing – Civilization

Tärtäria tablets

quipu

movable metal type, and composing stick, descended from Gutenberg's press

tortoise shells

cuneiform script
Language engineering is the art of creating languages.
Requestor Metaprogramming Abstractions

- EO Programming
  - Service collaborations
- VO Programming
  - Active variables (vars) composition
- VO Modeling
  - Model-driven VOP for multi-fidelity, multi-scale, multi-disciplinary collaborations

scripting, Java API, hybrid, visual
Service Composition

\[ f = f_1(f_2, f_3, f_4(f_5, f_6)) \]
task(
    sig("multiply", Multiplier.class),
    context(
        input("arg/x1", 10.0d),
        input("arg/x2", 50.0d)))
Job: service composition
f1(f2(f4, f5), f3)

Task f4 = task("f4", op("multiply", Multiplier.class),
context("multiply", input("arg/x1", 10.0d),
input("arg/x2", 50.0d), out("result/y1", null));

Task f5 = task("f5", op("add", Adder.class),
context("add", input("arg/x3", 20.0d),
input("arg/x4", 80.0d), output("result/y2", null));

Task f3 = task("f3", op("subtract", Subtractor.class),
context("subtract", input("arg/x5", null),
input("arg/x6", null), output("result/y3", null));

Job f1= job("f1", job("f2", f4, f5,
strategy(Flow.PAR, Access.PULL)), f3,
pipe(output(f4, "result/y1"), input(f3, "arg/x5")),
pipe(output(f5, "result/y2"), input(f3, "arg/x6"));
Types of Variables

• Variable (mathematics), a symbol that represents a quantity in a mathematical expression
• Variable (programming), a symbolic name associated with a value that may be changed
• Variable (OO programming), a set of object’s attributes accessible via ‘getters’
• Variable (SO programming), a triplet <value, evaluator, filter>
  – value: a valid quantity
  – evaluator: a service with dependent variables (composition)
  – filter: a getter
Basic Variable Structure (VFE)

\[ z = y_1(x_1, x_2, x_3) \]
Service Orientation

• A service: the work performed by a variable’s evaluator

• An evaluator defines:
  – Arguments (variable composition)
  – Processing services (mutifidelity)
  – Differentiation services (mutifidelity)
Type of Evaluator Services

• Command services (a)
  – Execute command (executables)

• Scripting services (a)
  – Execute expression (e.g., scripts, Java expressions)

• OO services (b)
  – Method invocation, RMI

• Federated services (c)
  – Federated method invocation (exertions)
    • Execute remote command
    • Execute expression remotely
    • Remote method invocation
    • Federated method invocation

Evaluators do realize other process expressions!—enable metaprogramming
Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and FSOOA
- SORCER metaprogramming and programming — EOL, VOL, VML
- SORCER Operating System (SOS) and FMI
- SORCER Virtual Processor and Provisioning
- Conclusions
Push vs. Pull execution

Pull execution allows for a pandemonium SO architecture.
Applying OO to Network (FMI)

- Service request is an object of type:
  \texttt{Exertion = Data Context + Signatures}

- Exertions are invoked by calling \texttt{exert}:
  \texttt{Exertion#exert(Transaction):Exertion}

- Exertions are executed by the network shell using collaborating service providers of type: \texttt{Servicer}
  - Service providers form P2P (S2S) environment
  - Service is requested by calling dynamically the \texttt{service} method
    \texttt{Servicer#service(Exertion, Transaction):Exertion}
  - A service provider is identified by the exertion signature’s \texttt{interface type} and optional attributes

- The signature operation \texttt{<operation>} is invoked on the matching service object:
  \texttt{public Context <operation>(Context) via Exerter#exert(Exertion, Transaction):Exertion}

The SORCER Triple Command Pattern
# UNIX Platform vs. SORCER Platform

<table>
<thead>
<tr>
<th></th>
<th>UNIX</th>
<th>SORCER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>File - file system</td>
<td>Data context - objects</td>
</tr>
<tr>
<td>Data flow</td>
<td>Pipes</td>
<td>Data context pipes</td>
</tr>
<tr>
<td>Cohesion</td>
<td>Everything is a file</td>
<td>Everything is a service</td>
</tr>
<tr>
<td>Processor</td>
<td>Native</td>
<td>Service providers</td>
</tr>
<tr>
<td>Interpreter</td>
<td>Shell</td>
<td>Exertion (network) shell</td>
</tr>
<tr>
<td>System language</td>
<td>C</td>
<td>Java/Jini/Rio/SORCER API</td>
</tr>
<tr>
<td>Command language</td>
<td>UNIX shell scripting</td>
<td>EO/VO/MD scripting</td>
</tr>
<tr>
<td>Process control</td>
<td>Command flow logic</td>
<td>Control context &amp; exertion flow logic (looping and branching)</td>
</tr>
<tr>
<td>strategy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executable codes</td>
<td>Many choices</td>
<td>Many choices</td>
</tr>
</tbody>
</table>

Unix pipes – processes; SORCER pipes – data contexts  
Pipeline vs. SORCER federation – exertion + control context + control flow exertions  
Local shell vs. network shell
Agenda

• Intro: computing science & process expression
• Distribution, object & service orientation
• Transdisciplinary computing processes — SO Platform
• C/S, SOA, SPOA, SOOA and FSOOA
• SORCER metaprogramming and programming
  — EOL, VOL, VML
• SORCER Operating System (SOS) and FMI
• SORCER Virtual Processor and Provisioning
• Conclusions
Exerting Dynamic Collaborations

Federation (Processor)

Service Collaboration Management (SOS)

SO Program (Requestor)

Signature type: preprocess process postprocess append
QoS and SLA

- **Quality of Service (QoS) Parameter**
  - a technical characteristic or performance benchmark of a resource

- **Service Level Agreement (SLA)**
  - a contract signed between a service requestor and a service provider for a specific time or task. It specifies that during the execution certain QoS parameters should maintain an agreed-on level or a fixed value.
Provisioning Component Diagram
## Provisioning Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Bootstrapping</th>
<th>Monitoring</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>manual</td>
<td>manual</td>
<td>no</td>
<td>till destroyed</td>
</tr>
<tr>
<td>manual-dynamic</td>
<td>manual</td>
<td>yes</td>
<td>till destroyed/configured</td>
</tr>
<tr>
<td>autonomic</td>
<td>auto</td>
<td>yes</td>
<td>till destroyed/configured</td>
</tr>
<tr>
<td>on-demand</td>
<td>auto</td>
<td>no</td>
<td>configured</td>
</tr>
<tr>
<td>on-demand-dynamic</td>
<td>auto</td>
<td>yes</td>
<td>till destroyed/configured</td>
</tr>
</tbody>
</table>
Agenda

• Intro: computing science & process expression
• Distribution, object & service orientation
• Transdisciplinary computing processes — SO Platform
• C/S, SOA, SPOA, SOOA and FSOOA
• SORCER metaprogramming and programming
  – EOL, VOL, VML
• SORCER Operating System (SOS) and FMI
• SORCER Virtual Processor and Provisioning
• Conclusions
Conclusions
SORCER FSOOA

• Discovery/join protocols
  – Location neutrality
• Service provider registration
  – Proxy object implementing service types
  – Proxy object owned by the provider
  – Proxy wire protocol(s) selected by provider
• Light-weight containers (service node, cybernode)
  – Small footprint JVM Hosting service providers
  – Static or dynamic deployment of service providers
  – Service assembly by DI
• OS (Tasker, Jobber, Spacer, Cataloger, Provisioner, Cybernode, ...)
  – FMI
  – Synchronous, asynchronous, QoS-optimized service federations
  – Provisioning
  – V1-V4 Enables Two Way Computing Convergence
  – SO shell
Q&A