Thinking Service Oriented Architecture (SOA)

Version 1.2, 1/18/2005
Revised 2/20/2005 & 1/28/2006

Authors

Michael Behrens, R2AD, LLC

Abstract

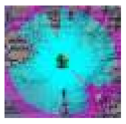
A paradigm shift requires a new way of thinking about a problem. Service Oriented Architectures (SOAs) represent a new approach to how systems are built. This paper helps one to start thinking in a SOA manner and makes a few recommendations on how to discover services and to make them resilient.

Status

This whitepaper is an initial draft release and is provided for review and evaluation only. The Companies hope to solicit your contributions and suggestions in the near future. The Companies make no warranties or representations regarding the specification in any manner whatsoever.

Thinking Service Oriented Architecture

If a minefield were available as overlay, then the graphics and perhaps a name would be displayable as part of the battlespace visualization in a C2 system. The operators that created the overlay would know what it meant and would include it on their briefs. With our net-centric hat on, that overlay could become available as a web service and made available to a wider community. However the meaning and importance of that shaded area would be lost outside the context in which it was created. A logical question is raised: should there be a minefield web service? It could provide the graphics (GML) as well as providing useful data about the minefield – the data behind the overlay (type, timing, status, country of origin, pedigree, etc). Taking this a bit further, higher order services would then use this service as they perform their functions such as route planning. A soldier in the field could ask the higher-order GIS route service to generate a route between two points. This route service could then invoke other sub-services for roads, minefields, blockades, checkpoints, enemy positions, sensors, weather, etc. and provide an appropriate route back to the soldier.

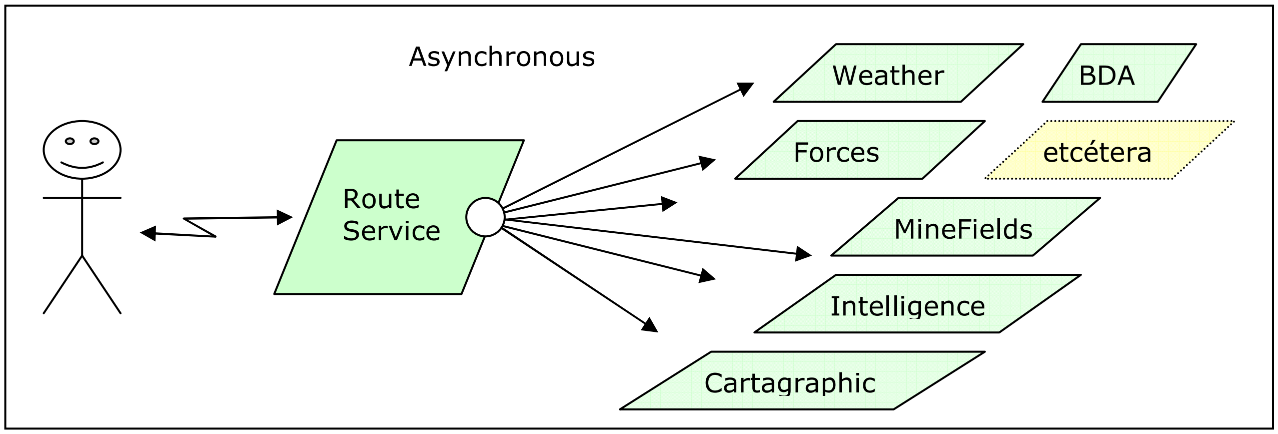


In general, everyone (architects, designers, PM) should think about how "data" can be used by others outside the enclave. The services should be self-describing, self-contained, and modular. Data should be capable of being externally referenced with some lifecycle guarantees. All these in turn can then be choreographed to make the network the computer.

Copyright © 2006 R2AD, LLC

Example Scenario

Consider an end-user on a thin client requesting for an evacuation route from a route service. Many back-end services are invoked automatically to ensure that the route provide is secure and fast. There is a hierarchy of services at work being invoked by the route service using the weighting factors provided by the end-user (i.e.: security is more important than speed). Some of these might include weather and the location of enemy and friendly forces. These



lower level services are also being invoked by other services independently in a stateless manner.

Many assumptions are made in the diagram above, however it is important to understand a few of them. Consider the minefield service for instance. It would need to expose a porttype which returns a list of minefields within a geographic bounds or along a multi-part path or corridor. Presumably, along with that information is an ability to obtain geo-rectified vector graphics for those minefields, in the case where the user is able to request visualization in a coordinate space.

Now suppose a new service is created which provides information about biological or chemical hazards. Can the system ingest that new data source and convey it as part of a command and control presentation service and be incorporated into the route service? Can this be done "without" any code changes to the existing route or presentation services?

If we can achieve that sort of dynamic data goal (which I believe is a worthy one), then we will succeed.

Service Registry

Any major system which is embracing SOA has a fundamental problem of communicating what services are available and more importantly, what services are being created or planned. Core services (such as those in NCES) provide lower order functions such as security auditing, messaging and storage. Applications (such as those in the JC2 space) create finer grain services.

This paper proposes a service registry (not UDDI) for Architects, System Engineers, and management/developers to go to in order to find out what all the current services are and also, what the planned services are. This would help reduce the wheel reinvention syndrome.

Therefore this registry would need to provide developer point of contact (POC) information, release schedules, draft and final (if fielded) WSDL files, and perhaps multiple endpoints which can be used for testing and for actual use.

Contracting agencies should be able to reference this registry to ensure that developers are not reinventing the wheel. Developers need to use this registry in their design and proposal documentation.

UDDI (and like) registries should update the Service Registry with current fielded implementation information.

Data Exchange

Selected data from the graph must be capable of being put into a clipboard or moved into other application components. Cut/Paste and Drag/Copy or Drag/Move operations are typically used in client applications and a similar model is expected to provide the bulk of this exchange.

WS Efficiency

Web services are an eventual replacement for language specific APIs which are currently bound to a specific programming language. Instead of binding to a Java or C++ library in order to convert from MGRS to Lat/Lon, there would be an available network service call. However the question granularity arises which is most likely directly proportional to overall system efficiency. Eventually, language specific wrappers (APIs) are created to allow easy interaction with the services. Services will soon provide caching as well, to increase efficiency. Services bandwidth concerns will cause a wrapper around the network layer to make them condensed, essentially creating a CORBA like protocol equivalent. So in the end, the Model Driven Approach shall be the model to follow whereby from the model, a plethora of language specific bindings to network behavior can emerge.

Mobile Services – Grid Bound

A logical next step is to consider how web services can become mobile. This is similar to what the Grid world offers as part of the concept of virtual organizations or virtual enterprises.[1] Mobile services can be thought of as mobile code, however they expose a network interface on the target platform. They can be cloned and spread around a LAN, providing scalable survivable services which are dynamically registered by the Service Container.

Acknowledgements

References

[“WS Orchestration”, a review of emerging technologies, tools, and standards, Jan 03]
http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf

[Ian Foster1,2 Carl Kesselman3 Jeffrey M. Nick4 Steven Tuecke1, "The Physiology of the Grid"]
http://www.globus.org/research/papers/ogsa.pdf

[1] Open Grid Services Architecture (OGSA)

Copyright © 2006 R2AD, LLC

© 2005 R2AD, LLC Corporation. All rights reserved. No other rights are granted by implication, estoppel or otherwise.

R2AD is a registered trademarks of R2AD, LLC in the United States and/or other countries.
The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Permission to copy and display this "Thinking SOA" Whitepaper ("this Whitepaper"), in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of this Whitepaper, or portions thereof, that you make:

1. A link or URL to this Whitepaper at this location.
2. This Copyright Notice as shown in this Whitepaper.

The information contained in this document represents the current view of R2AD, LLC Corporation on the issues discussed as of the date of publication. Because R2AD must respond to changing market conditions, it should not be interpreted to be a commitment on the part of R2AD, and R2AD cannot guarantee the accuracy of any information presented after the date of publication.

R2AD, LLC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

R2AD may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from R2AD, LLC, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). THIS WHITEPAPER IS PROVIDED "AS IS". R2AD, LLC, BiblioTronix, LLC (COLLECTIVELY, THE "COMPANIES") MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS WHITEPAPER ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE COMPANIES WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS WHITEPAPER.