



TIBCO[®]
The Power of Now[®]

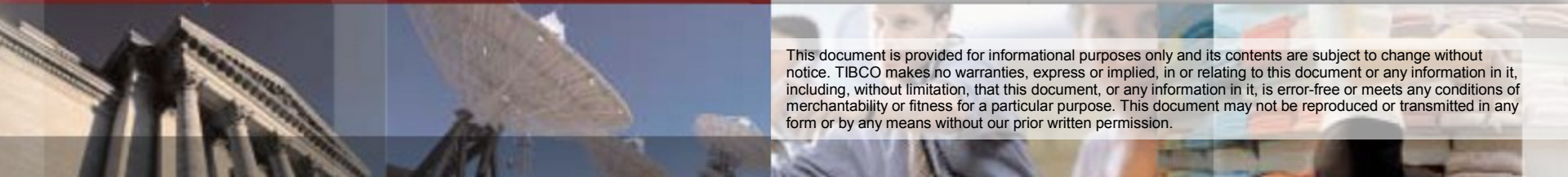


The KISS Principle Applied to SOA Using SCA

**How can all those SCA
specifications make my life
easier!?**

Eric Johnson

Principal Architect, TIBCO Software Inc.



This document is provided for informational purposes only and its contents are subject to change without notice. TIBCO makes no warranties, express or implied, in or relating to this document or any information in it, including, without limitation, that this document, or any information in it, is error-free or meets any conditions of merchantability or fitness for a particular purpose. This document may not be reproduced or transmitted in any form or by any means without our prior written permission.



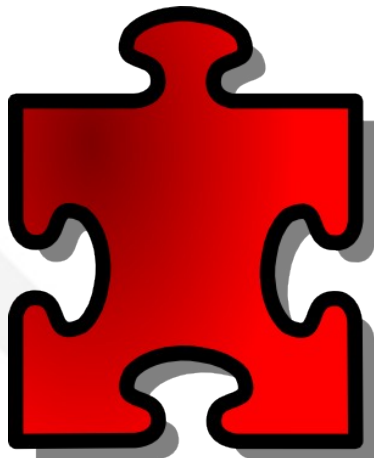
Enterprise Software – where are we now?

Complexity, what complexity?

SAML WS-Security WS-I BasicProfile
Java WS-Transactions SOAP 1.1
XACML WS-MetadataExchange UDDI
Atom WS-Policy .NET
SMTP HTTP **SCA** SOAP 1.2
XML REST OpenID
WSDL 1.1 BPMN WS-ReliableMessaging
BPEL JSP WSDL 2.0 RelaxNG
JMS XML Schema EJB

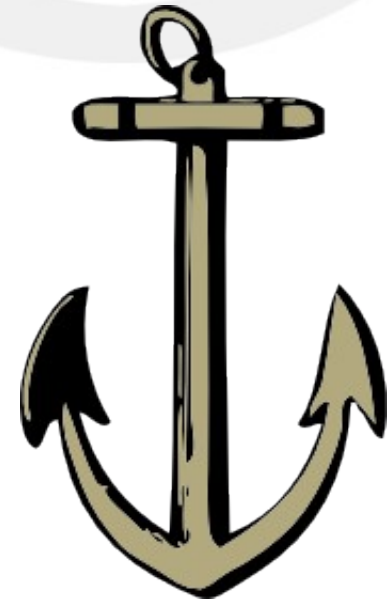
What are the problematic details of enterprise software?

- **Very complex (read – expensive)**
- **Concepts vary from vendor to vendor**
 - even from tool to tool within vendors!
- **People forced to make a significant investment based on particular vendors tools**
- **Tied to a platform, it is difficult to refactor and reuse business logic**



Abstractions are essential, if they're the right ones.

- Java EE solves lots of problems by pushing programming problems to the “container”.
- Brought together technologies in a pre-“services” world.
- Now we need something for the services world...

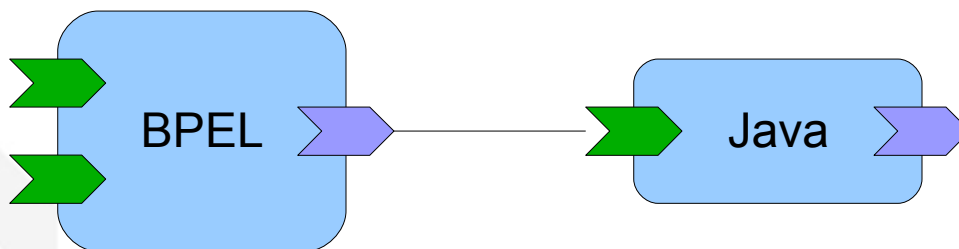




Is it going to get better?

Yes! SCA means to help! And here's how...

- Defining common concepts for design, implementation, configuration, and deployment
- Clear(er) separation of roles
- Isolating business logic from environment
- Separates design “intent” from how functionality gets exposed



SCA brings together all of your SOA related technologies into one context.

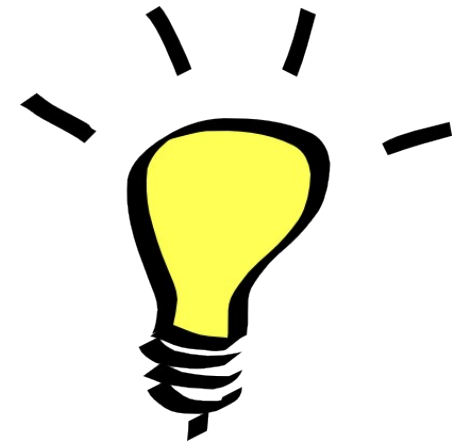
- Unifies how business logic exposes itself via services, and accesses other services via references
- Less vendor lock-in over time
- ... sounds great – so what's the catch?

There are just a few gotchas lurking around SCA.

- Fully extensible set of specifications
- Not fully baked – OASIS TCs still in progress
- Vendor support?

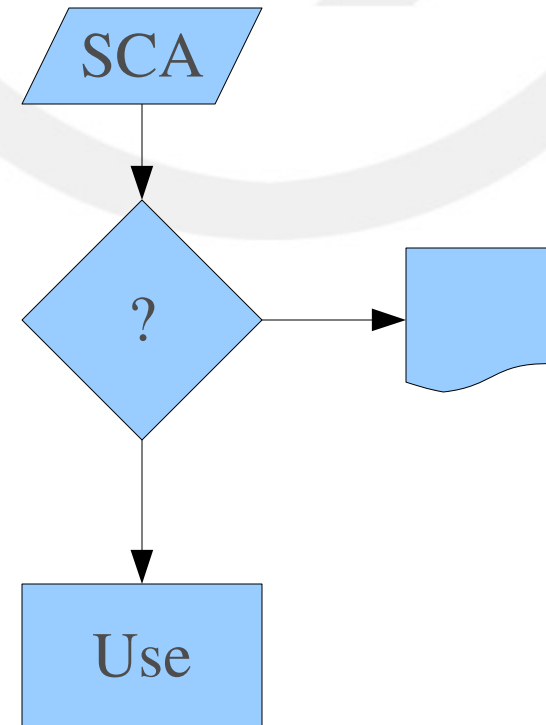


**SCA has clear benefits, but brings new complexity.
Now what do I do?**



What techniques make sense to apply to using SCA to keep the problems tractable?

- Avoid the full of complexity of the specification
- Develop centers of expertise, and start small
- Establish conventions
- Stick to a single vendor
- Maybe even *ignore* the vendors
- Above all, apply the KISS principle



Avoid the full complexity of the specification.

- Pick parts that will help you out *right now*
- Use it for a particular purpose
- Follow expected paths



Develop centers of expertise and start small.

- **Pick a core group of people that will start with SCA**
- **Use this group to figure out what is appropriate**
- **Invest in exposing them to a variety of sources of information and training**
- **Have additional teams start using SCA only after core team is ready to mentor**



Establish clear conventions for use.

- **Even better – use conventions set out by a vendor's tool**
- **Validate conformance to those conventions**
- **Examples:**
 - Always/never assign policy intents to references
 - Use only agreed-upon implementation and binding types
 - Never use “wires” stand-alone
 - Use only single-part WSDL messages



Stick to a single vendor.

- **Do you constantly switch database, compiler, and operating system vendors...?**
- **Pressure the vendor of your choice to satisfy your needs**
- **Learn the specifics of the vendor's tools, exploit them when appropriate.**

What do you mean by “ignore the vendors!?” How can that work?

- **Maybe you think it is too early to get on the bandwagon?**
- **Options:**
 - Start learning the specifications – learn the terms, concepts, patterns, and approaches
 - Apply SCA ways of thinking in your architecture
 - Use it like you would UML – diagram your services
- **... and you're a lot further along when the vendors catch up to you.**

Continually reassess how you measure up to the KISS principle.

- SCA, BPEL, SDO, JAX-WS, SOAP, WS-ReliableMessaging, JMS, WSDL – do you really need it *all*?
- ***Unnecessary complexity is the enemy of***
 - Predictability
 - Reliability
 - Security

Conclusion

- **Get on the SCA bandwagon**
- **Do so cautiously**
- **... and of course, I think you should use TIBCO products to do so**



Questions?



Credits

- **Original SCA work: <http://www.osoa.org>**
- **Ongoing work at OASIS: <http://www.oasis-open.org>**
- **Artwork from <http://www.openclipart.org>**
 - converted to PNG with Inkscape (<http://www.inkscape.org/>)
 - Puzzle piece: <http://openclipart.org/media/files/nicubunu/7859>
 - Anchor: http://openclipart.org/media/files/johnny_automatic/2914
 - Dragon: <http://openclipart.org/media/files/PeterM/114>
 - Lightbulb: <http://openclipart.org/media/files/Anonymous/7121>
 - Road: <http://openclipart.org/media/files/abadr/7686>
 - People at table: <http://openclipart.org/media/files/neocreo/5282>
 - Book: <http://openclipart.org/media/files/barretr/6170>