Architecture AntiPatterns
Part 2

architecture anti-patterns 2
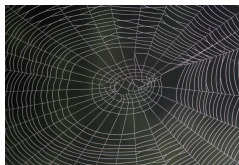

armchair architecture

infinity architecture

playing with new toys

groundhog day

spider web architecture

stovepipe architecture

armchair architecture

whiteboard sketches are handed off as final architecture standards without proving out the design

armchair architecture

occurs when you have non-coding architects

occurs when architects are not involved in the full project lifecycle

occurs when architects don't know what they are doing

armchair architecture

stay current and try to carve out some coding for yourself, even if it is proof-of-concept code

the best architects are the ones who have been in the trenches themselves and know the fallout from bad architecture decisions

be careful not to release architecture decisions and standards too early

armchair architecture

be an integral part of your development team!!

playing with new toys

incorporating unproven technologies into an architecture that don't really fit the problem at hand

playing with new toys

when introducing a new technology, ask yourself:

purpose: what value is it delivering?

proven: is this a proven technology for your situation?

overlap: is there something we have that is already supplying this functionality?

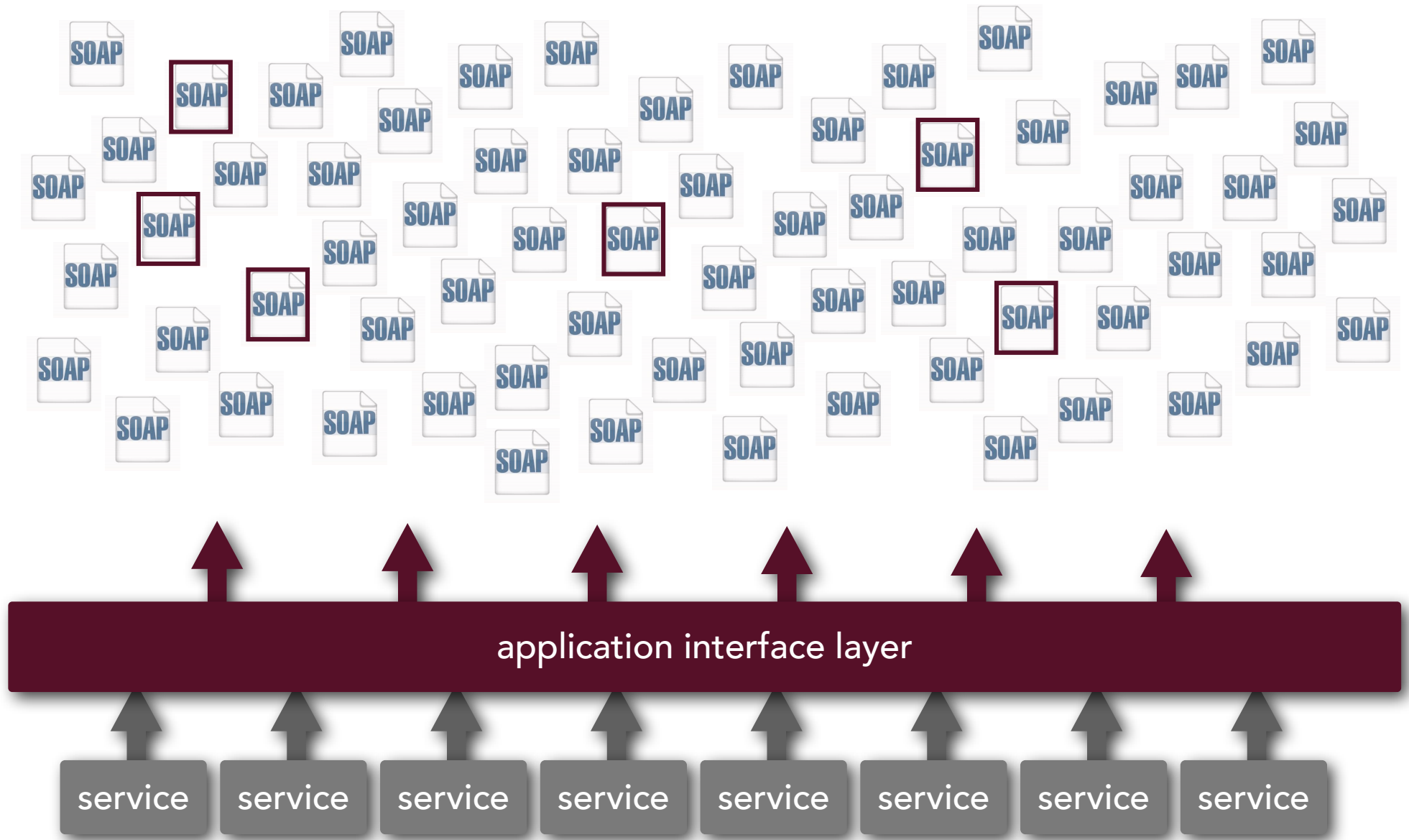*feasibility*: does your team have the skills necessary for the technology?

spider web architecture

creating large numbers of web services that
are never used just because you can

spider web architecture

application interface layer

service service service service service service service service

spider web architecture

just because you can create a web service at the click of a button doesn't mean you should!

let the requirements and business needs drive what services should be exposed

infinity architecture

creating architectures and interfaces that are over-generalized with infinite flexibility
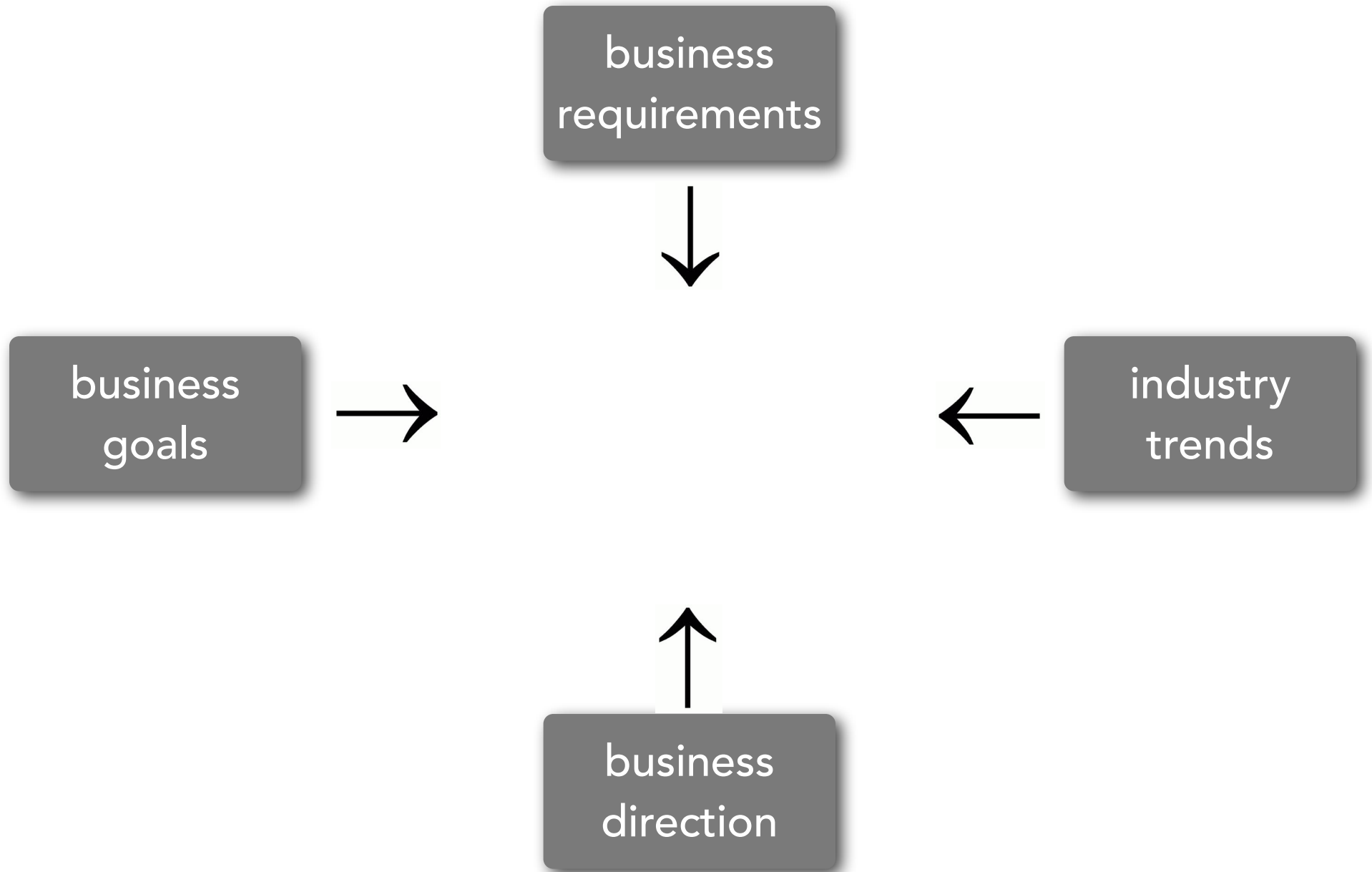
infinity architecture

generalized architectures that solve every possible need are expensive and difficult to maintain and change - "we may need..."

instead, use domain-specific architectures to reduce architecture and system scope
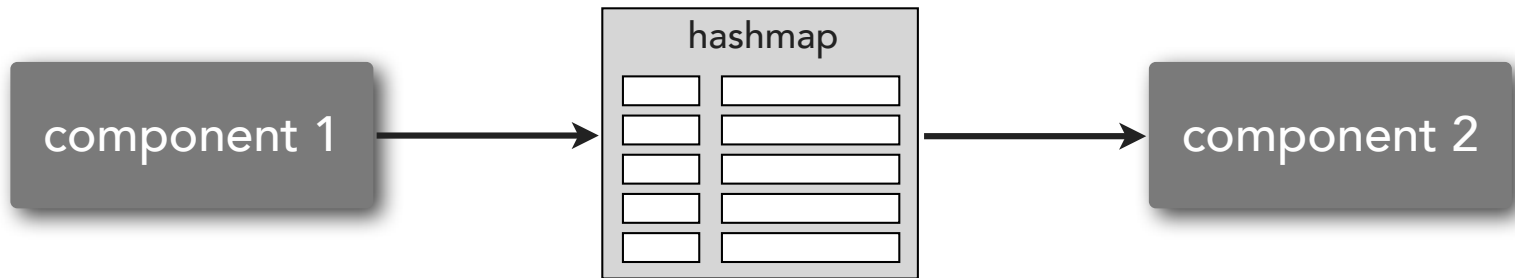
infinity architecture

infinity architecture

generalized interfaces between components provides infinite flexibility, but at a cost....

great, but where's the contract?

groundhog day

critical architecture decisions made early on are lost, forgotten, or not communicated effectively

groundhog day

symptoms and consequences

people forget or don't know a decision was made

the same decision keeps getting discussed and made over and over and over…

no one understands why a decision was made and they begin to question it again and again…

groundhog day

avoidance techniques

capture all important architecture decisions in some sort of work product (doc, wiki, etc.) and make it centrally available

make sure the right stakeholders know about critical decisions and where to find them

stovepipe architecture

an ad-hoc collection of ill-related ideas, concepts, and components that leads to a brittle architecture

stovepipe architecture

symptoms and consequences

lack of proper abstraction

lack of an integration solution

lack of architecture guidance

architectures that are difficult to change, difficult to maintain, and break every time you change something

stovepipe architecture

related anti-patterns

architecture by implication

witches brew

stovepipe architecture

?'s

Mark Richards
Independent Consultant
Hands-on Enterprise / Integration Architect
Published Author / Conference Speaker

http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9

Published Books:
Java Message Service, 2nd Edition
97 Things Every Software Architect Should Know
Java Transaction Design Strategies

Neal Ford
Director / Software Architect /
Meme Wrangler

ThoughtWorks®

2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1   40 4242 9929   Twitter: @neal4d
E: nford@thoughtworks.com   W: thoughtworks.com