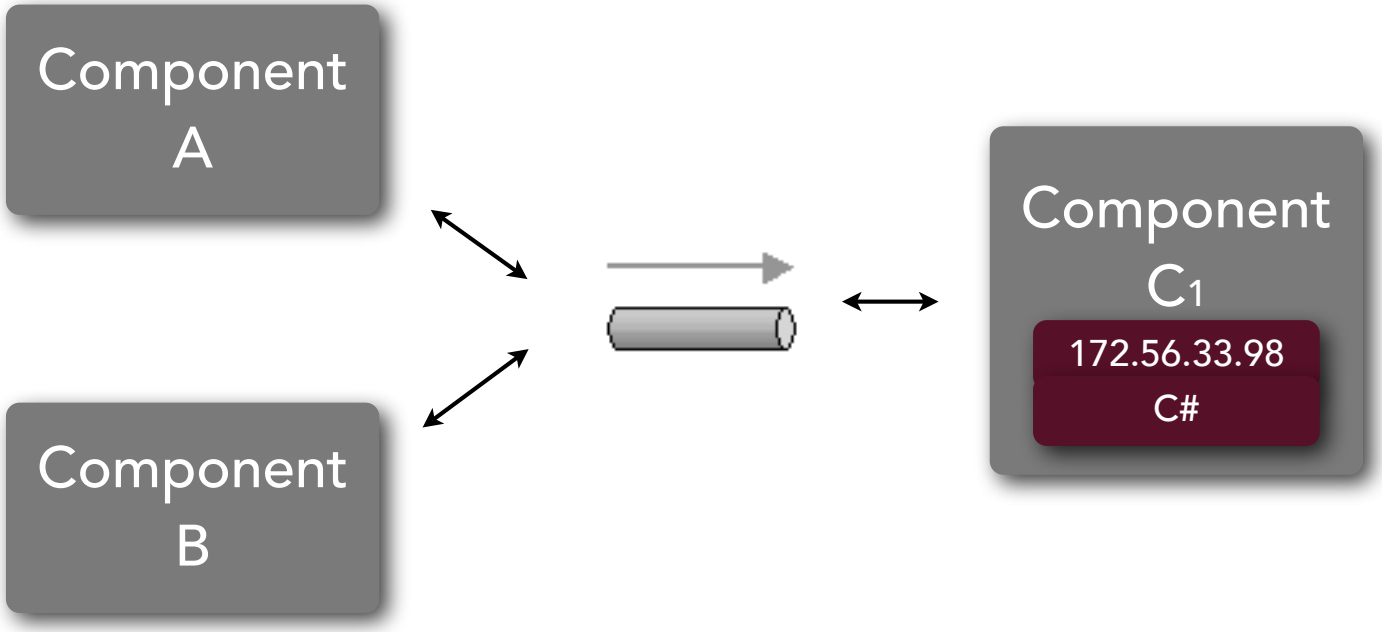
Applying Abstraction

applying abstraction

abstraction minimizes change



applying abstraction

but it comes with a price...

decreased performance

added complexity

increased development and testing effort

increased maintenance effort

applying abstraction

the extent to which you apply abstraction varies based on the tradeoffs you are willing to accept

aspects of abstraction

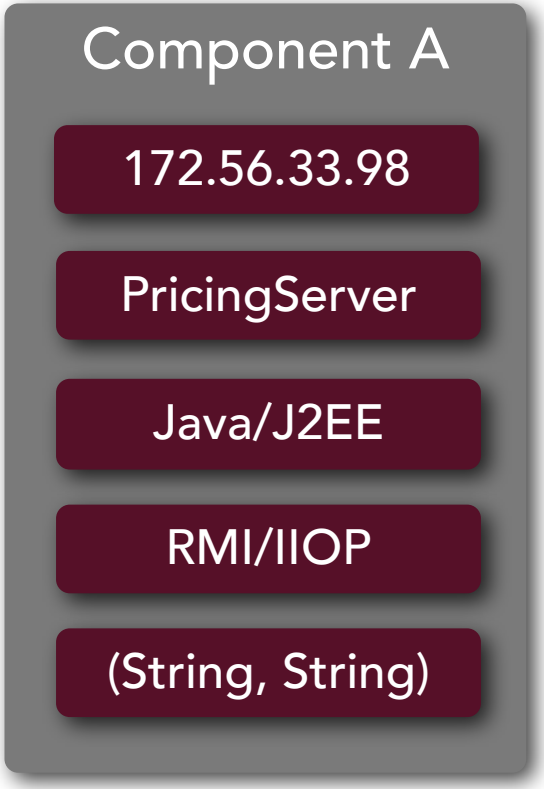
location transparency

name transparency

implementation transparency

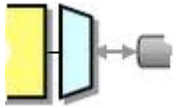
access decoupling

contract decoupling

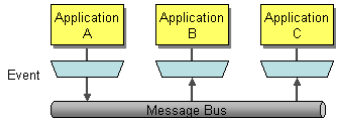


methods of abstraction

messaging

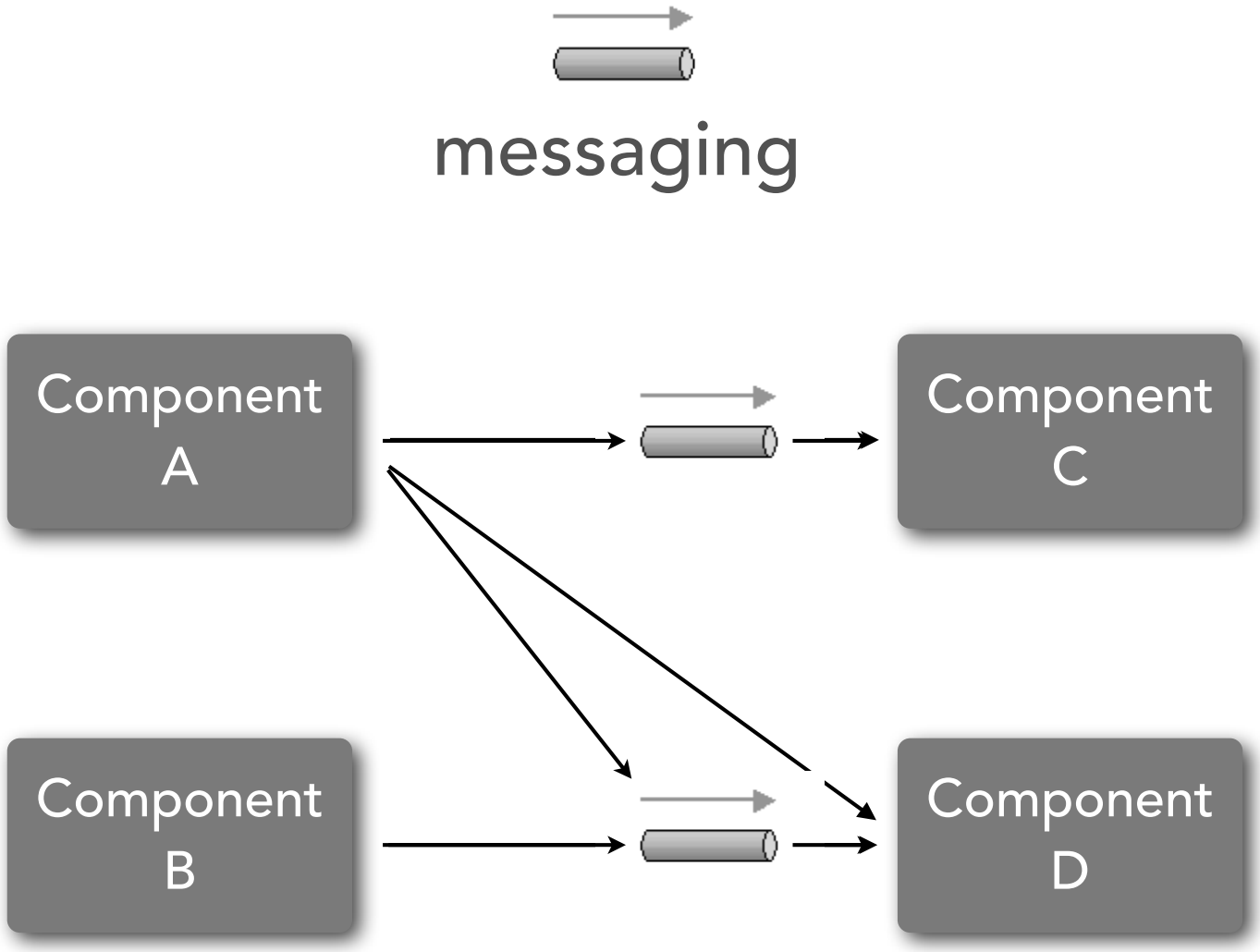


adapter



message bus

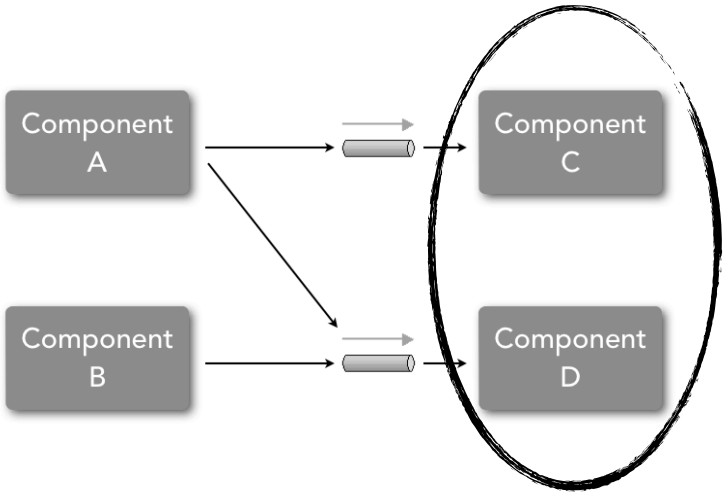
methods of abstraction



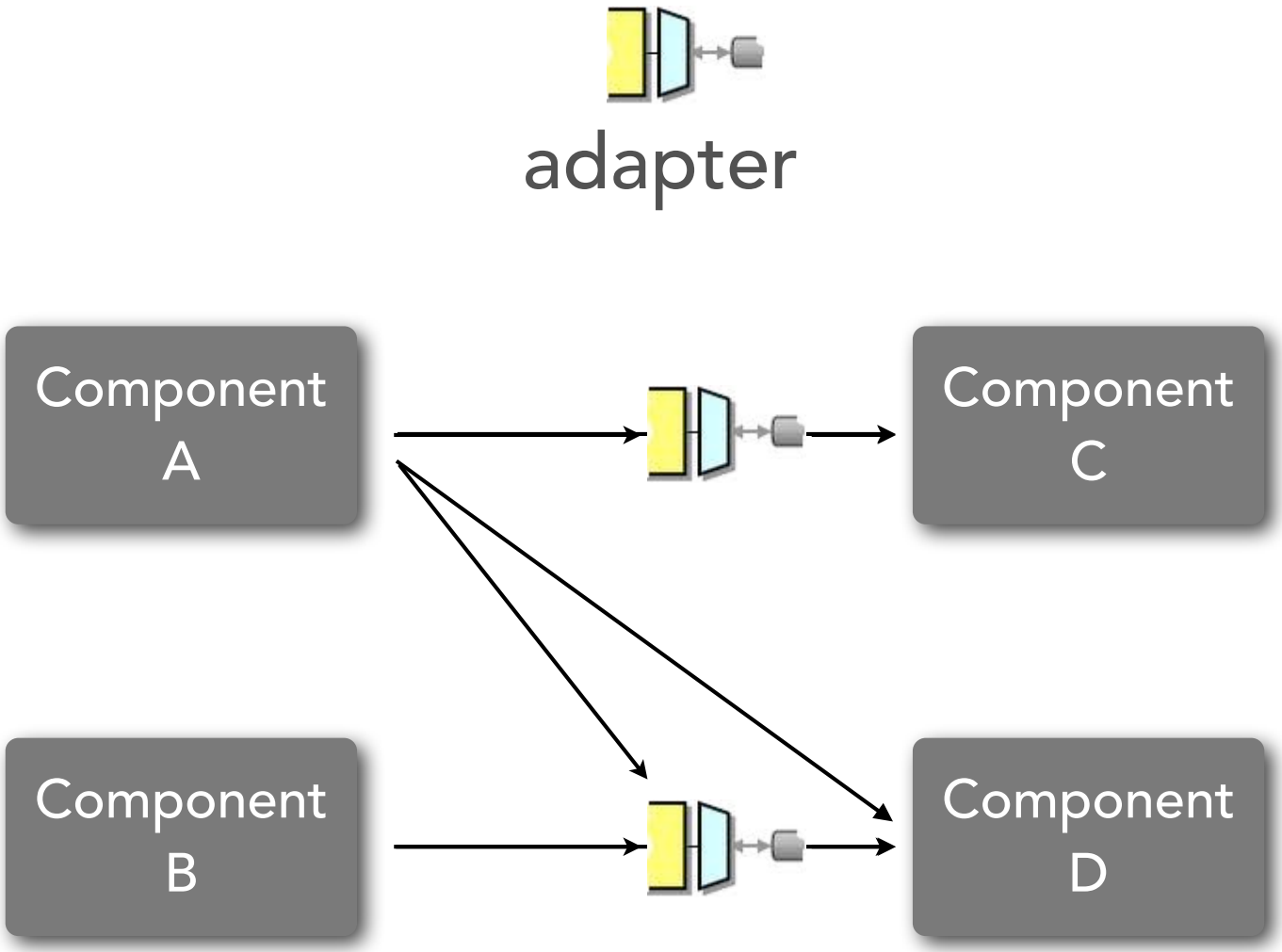
methods of abstraction

messaging

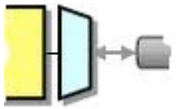
- location transparency
- name transparency
- implementation transparency
- access decoupling
- contract decoupling



methods of abstraction

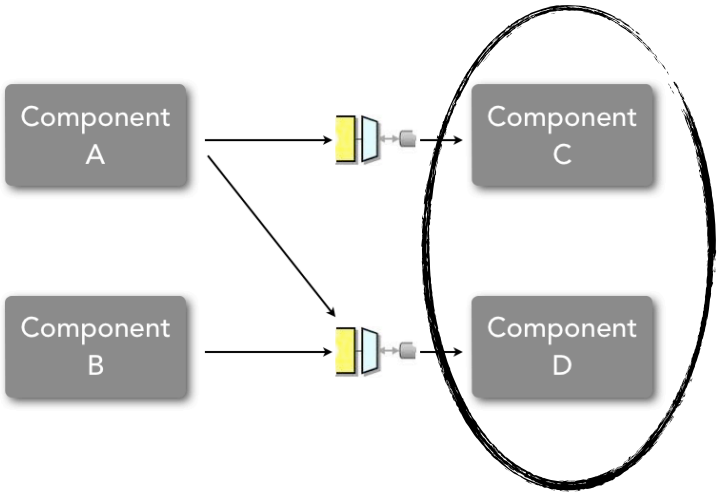


methods of abstraction

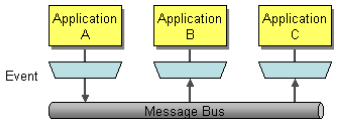


adapter

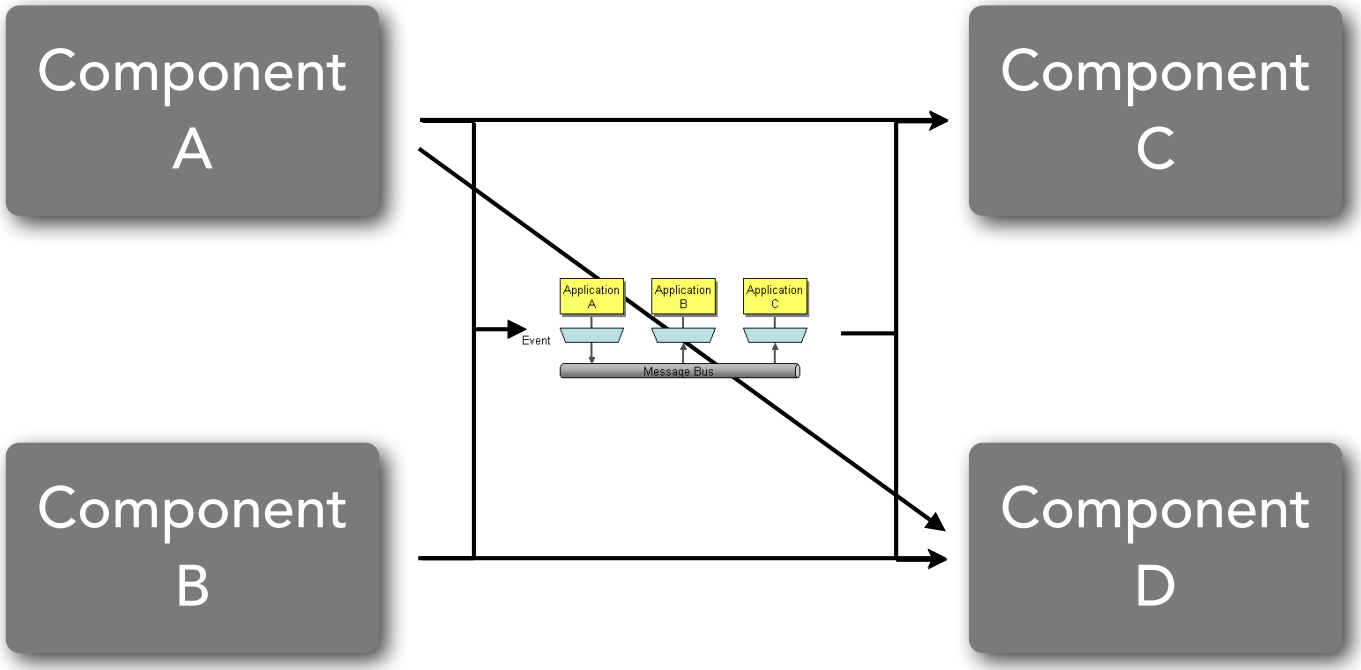
- location transparency
- name transparency
- implementation transparency
- access decoupling
- contract decoupling



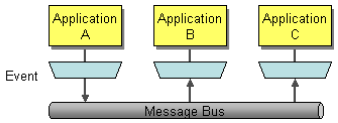
methods of abstraction



message bus

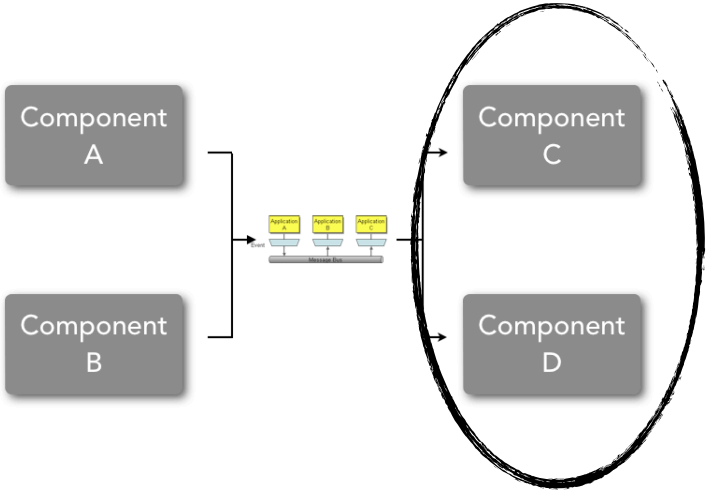


methods of abstraction



message bus

- location transparency
- name transparency
- implementation transparency
- access decoupling
- contract decoupling



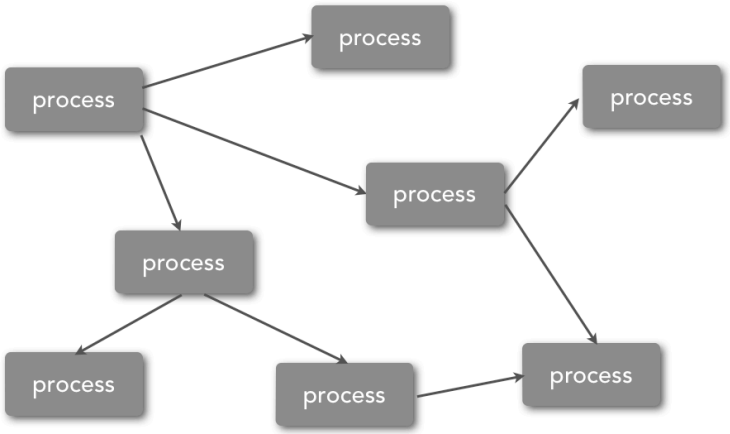
abstraction patterns

layered architecture

each layer of the architecture is isolated
from changes made at different layers

abstraction patterns

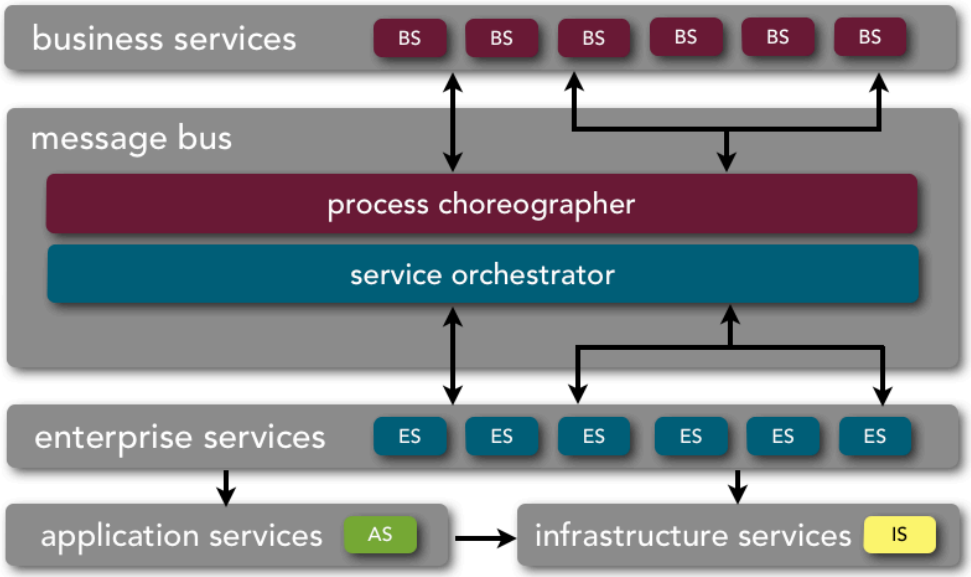
event-driven architecture



each component acts as an independent processor for a specific event and is fully decoupled from other processors

abstraction patterns

service-oriented architecture



abstraction is achieved by separating business services from implementation services and through the use of a message bus

?'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