Incorrect SOA Assumptions Can Lead to Mission Disaster

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

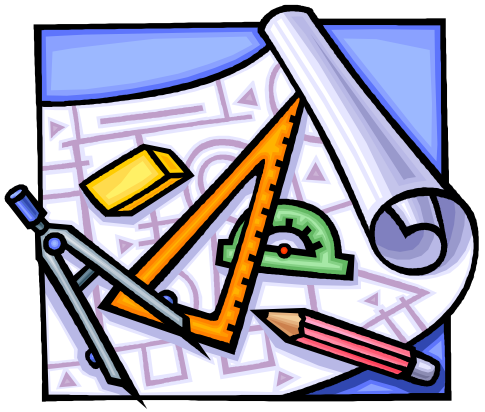
Grace Lewis, Edwin Morris, Soumya Simanta, Lutz Wrage

Software Engineering Institute | Carnegie Mellon

What is SOA?

Service-oriented architecture is a way of designing systems that enables

- Cost-efficiency
- Agility
- Adaptability
- Leverage of legacy investments



Services

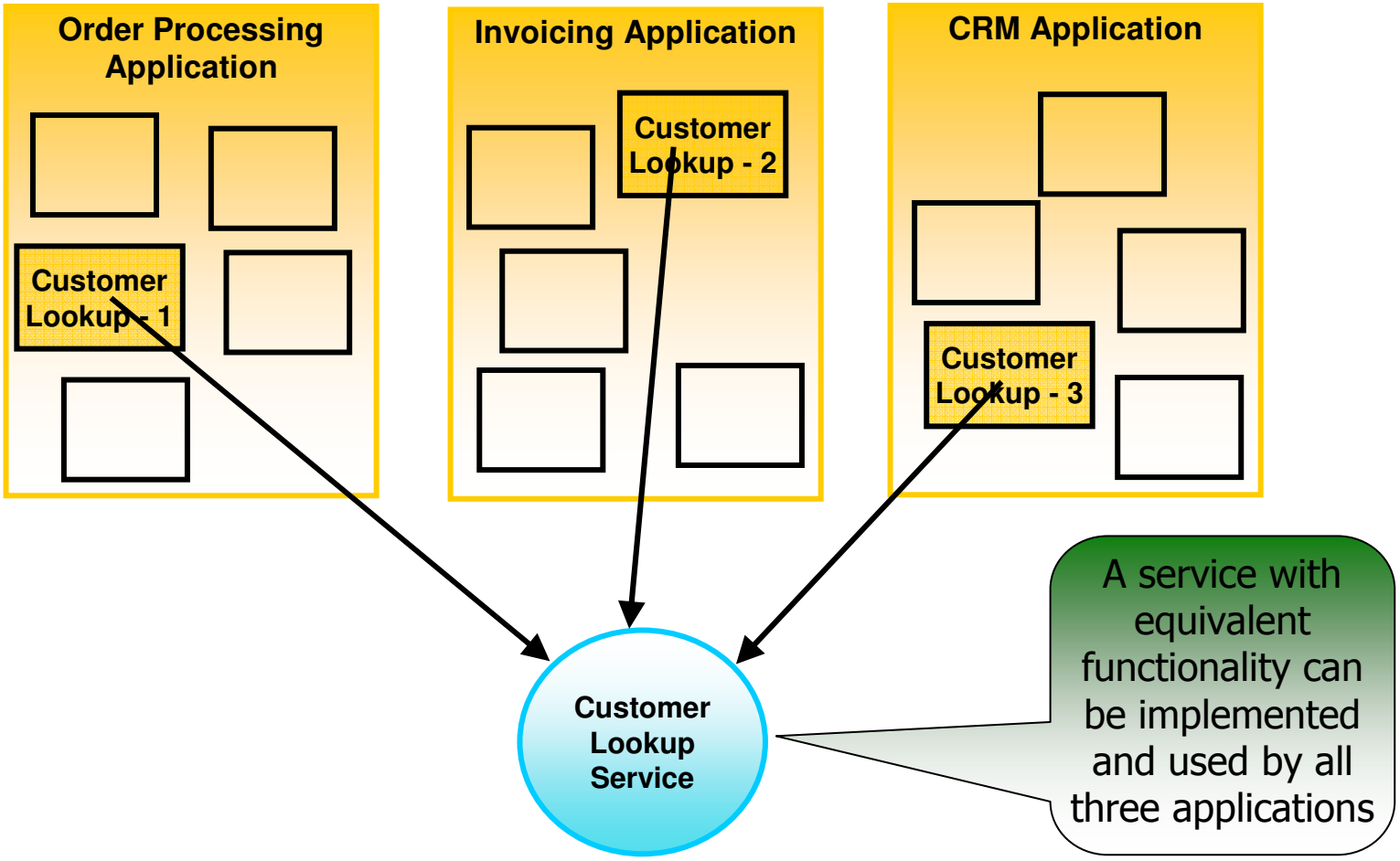
Services are reusable components that represent business tasks.

- Customer lookup
- Account lookup
- Credit card validation
- Weather
- Hotel reservation

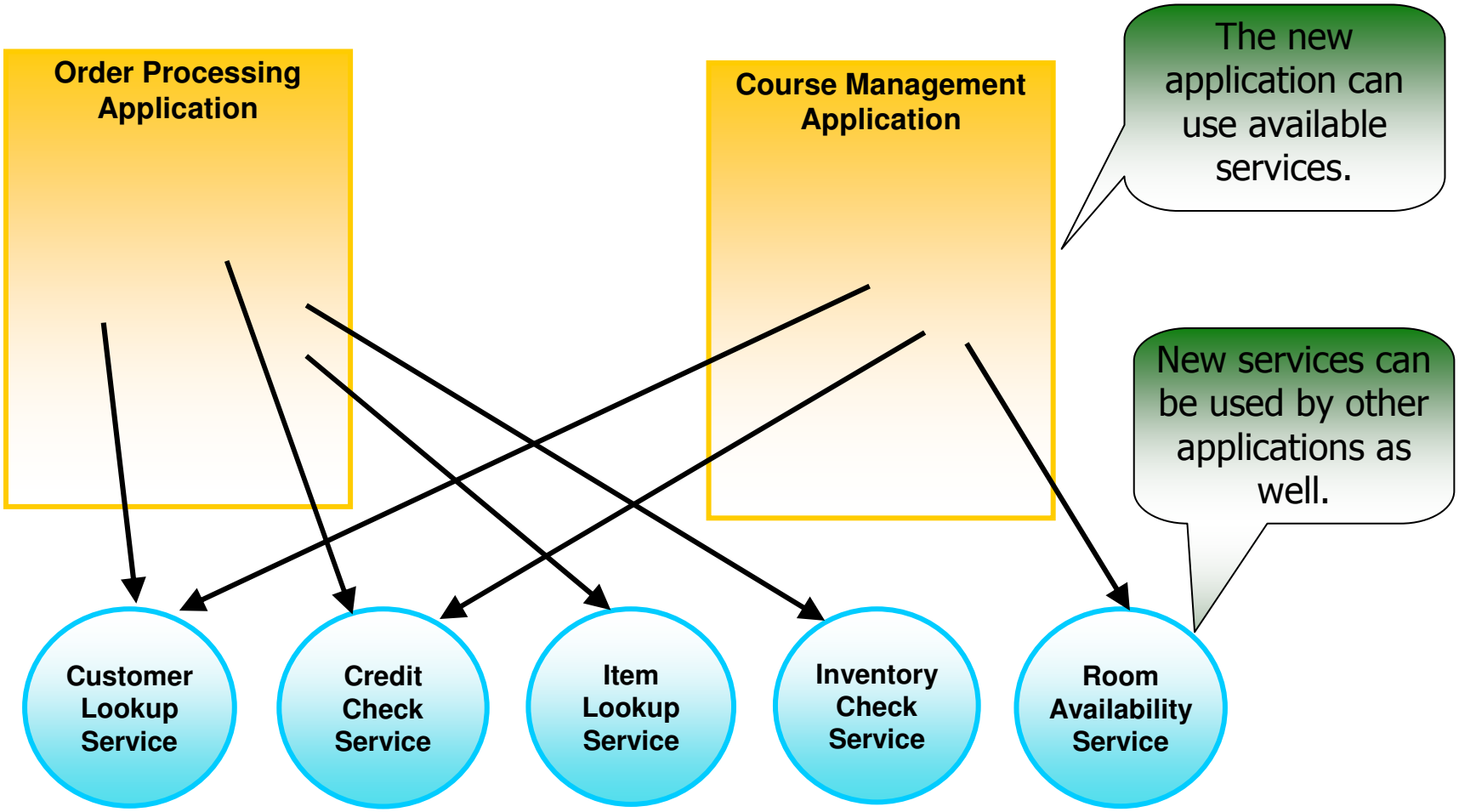
Services can be

- Globally distributed across organizations
- Reconfigured into new business processes

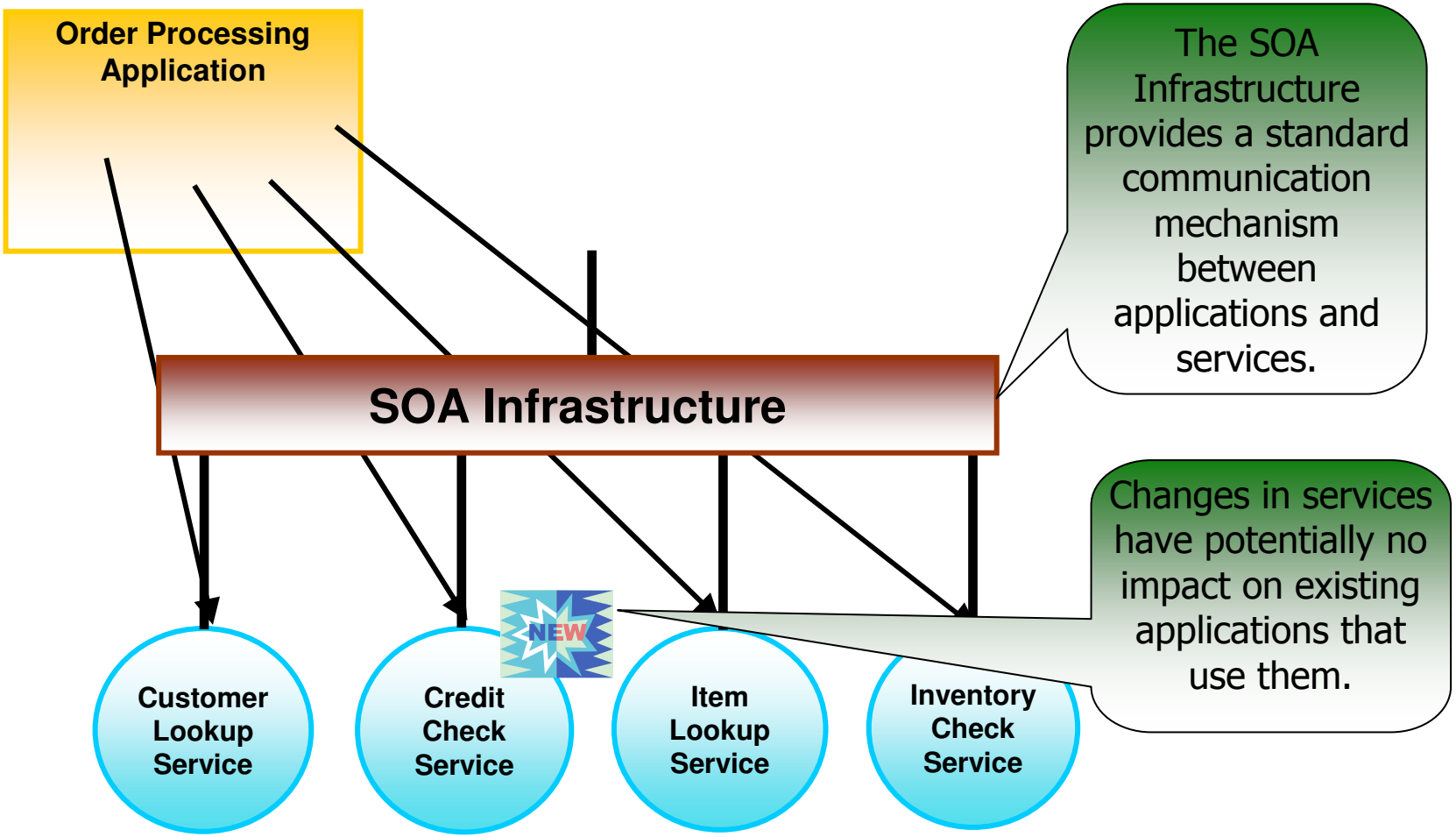
Services and Cost-Efficiency



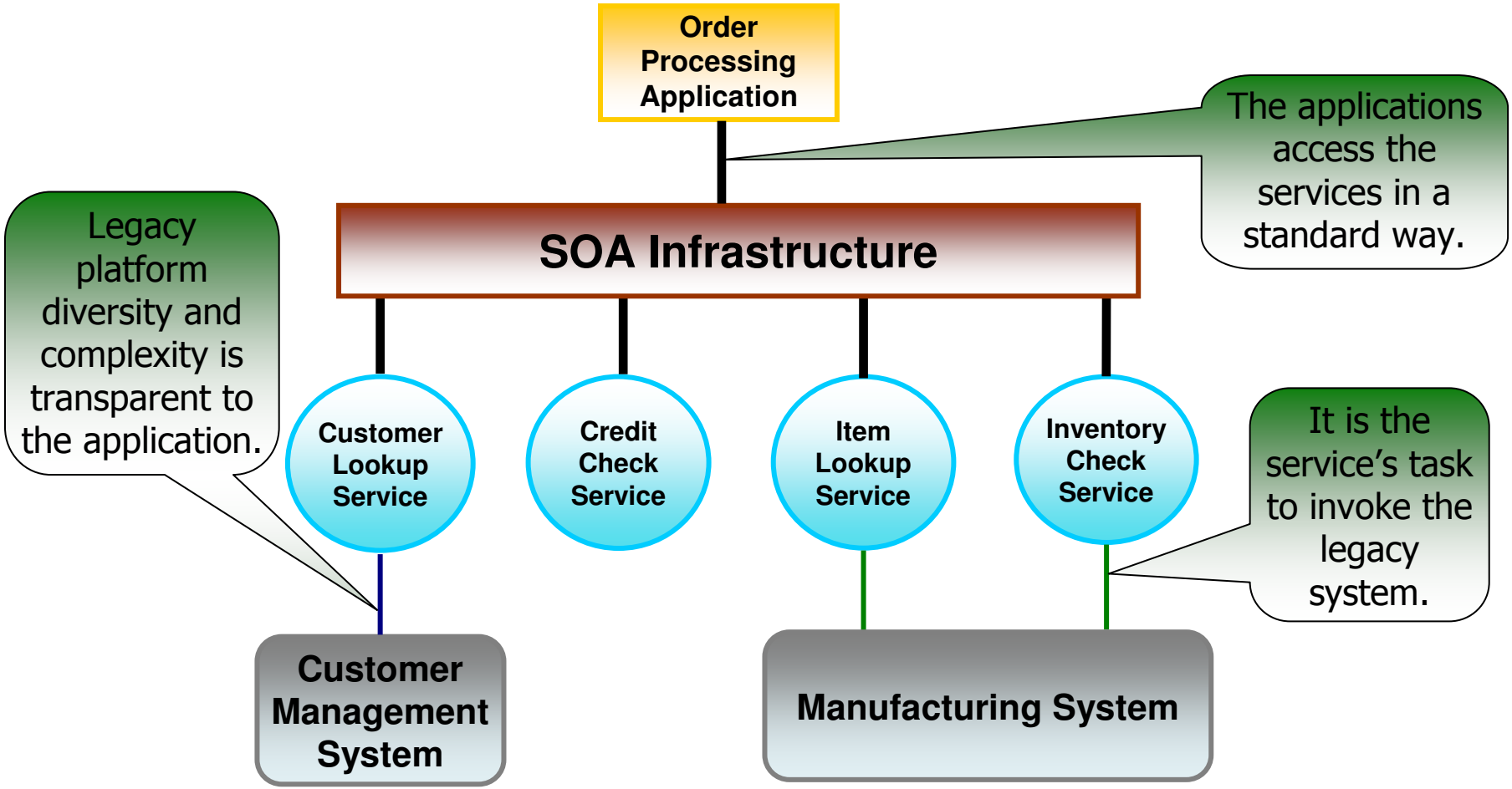
Services and Agility



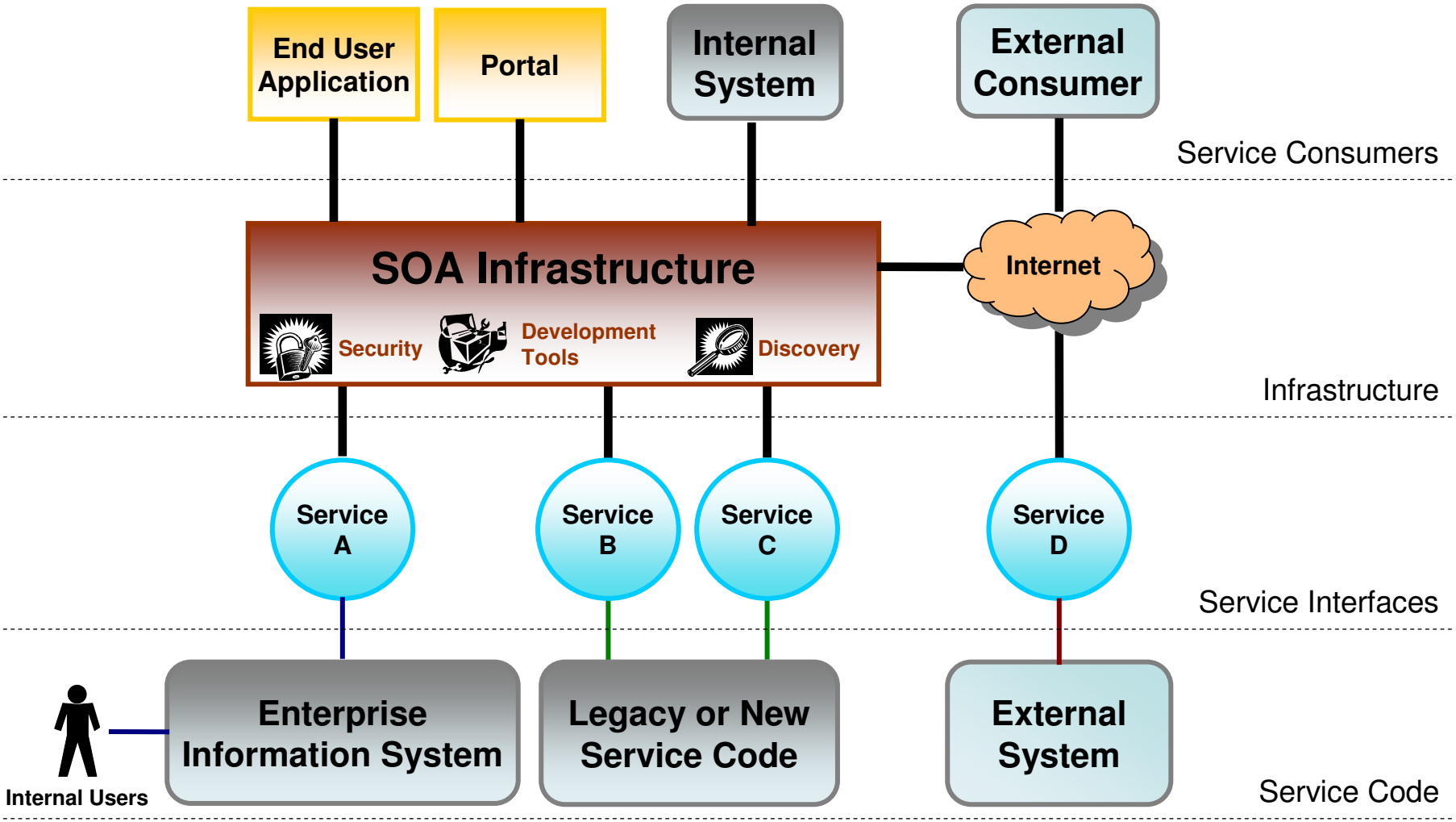
Services and Adaptability



Services and Legacy Leverage



Components of an SOA-Based System



In Summary …

SOA is an approach to software development where

- Services provide reusable functionality with well-defined interfaces
- An SOA infrastructure enables discovery, composition and invocation of services
- Applications are built using functionality from available services

If managed well, SOA adoption can lead to

- Cost-efficiency
- Agility
- Adaptability

However, there are some misconceptions associated with SOA …

SOA Provides The Complete Architecture For A System

SOA is an architectural pattern/style/paradigm and not the architecture of the system itself

An architectural pattern provides guidance that embodies best practices

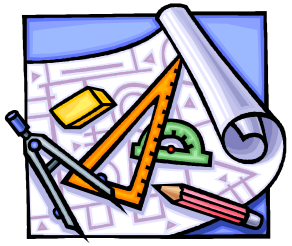
- The concrete elements and their interactions are the architecture of the system

Any number of systems can be developed based on an architectural pattern

- An architecture based on SOA inherits both the good and the bad

Corollary: SOA cannot be bought off-the shelf

- System qualities have to be built into the architecture of the system
- Decisions have to be made—service design and implementation, technologies, tradeoffs



All Legacy Systems Can Be Easily Integrated Into An SOA Environment

Upfront hands-on analysis on the technical feasibility and return on investment must be performed to avoid last minute surprises

- Is it technically feasible to create a service from the legacy system or part of the system?
- How much would it cost to expose the legacy system as services?
- Is this cost plus the cost of maintaining the legacy system more than the cost of replacing it with a new one?
- What changes will have to be done to the legacy system?
- How much will these changes affect current users and other production systems?

It might just not make sense to migrate the legacy system to an SOA environment

SOA Is All About Standards and Standards Are All That Is Needed

"The good thing about standards is that there are so many to choose from"

Comes from misconception that SOA and Web Services are the same

Most Web Services standards are emerging—subject to multiple interpretations

- Basic standards are quite stable: WSDL, SOAP, XML
- Higher-level standards are immature / under development
- There are (too) many standards—overlaps and potential conflicts require careful evaluation
  - BPEL vs. WSCL vs. WS-Coordination
  - SAML vs. WS-Security

The Use of Standards Guarantees Interoperability in an SOA environment

Interoperability needs agreement on both syntax and semantics

Web Services enable syntactic interoperability

- XML Schema defines structure and data types
- WSDL defines the interfaces: operations, parameters and return values
- Available information, technologies and tool support

Web Services do not guarantee semantic interoperability

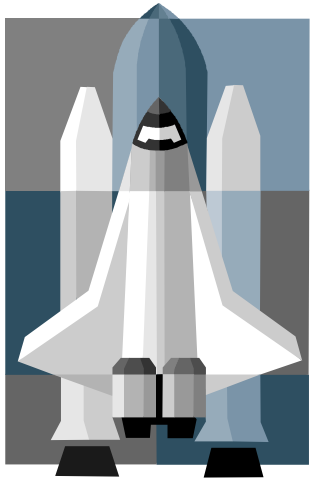
- XML and WSDL do not define the meaning of data
- WSDL does not define what a service does
- Active research area—unresolved issues

SOA Is All About Technology 1

SOA not only means a shift in technology but also changes in the organizational governance model

SOA governance provides a set of policies, rules, and enforcement mechanisms for developing, using and evolving SOA assets, and for analysis of their business value.

- Policies and procedures
- Roles and responsibilities
- Design-time governance
- Runtime governance



SOA Is All About Technology 2

What life-cycle model should be followed for services?

- Service requirements and definition
- Service development, composition and testing
- Service evolution and change management

What other mechanisms are required?

- Conflict resolution
- Deployment mechanisms
- Monitoring mechanisms
- Enterprise-wide policies
- Service-level agreements
- Service repositories

It Is Very Easy To Develop Applications Based on Services

It is relatively easy to build applications and services that work with a particular infrastructure … but designing a "good" service might not be that easy

From a service provider perspective

- Not many best practices for designing services
  - What is the right granularity?
  - What is the right Quality of Service? Can you guarantee it?
- Have to know and anticipate potential consumers and usage patterns
  - "If you build it they will come" – Can you afford this?

From a service consumer perspective

- Ease depends on tool availability for SOA infrastructure
- Larger granularity may lead to larger incompatibilities
- Most difficult part is composition—data mismatches

A Service Registry Allows Service Binding Dynamically at Runtime

Current technologies have not advanced to the point that this is possible in production environments

Requires the use of a common ontology by service providers and consumers within a domain

Requires the construction of intelligent applications that

- Construct the right queries for the discovery of services
- Compose services when there is not a single service that can process the request
- Provide the right data to invoke a service that was discovered at runtime

Testing SOA-Based Systems Is No Different than Testing Any Other Type of System

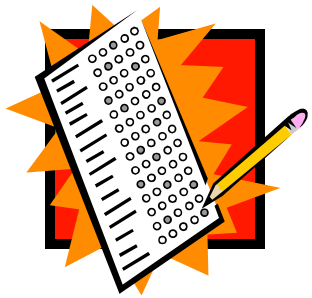
Testing service consumers, as well as the services themselves, is challenging for various reasons

For service consumers

- End-to-end testing is only possible when the invoked services (or test instances of them) are available
- Service consumers will necessarily have to be prepared to deal (or not to deal) with degraded service modes and complete service failure
- QoS may vary with the load on the network and on the service

For service providers

- Testing has to be based on anticipated usage patterns and scenarios
- Testing for satisfaction of SLAs is challenging
- Changes can potentially affect consumers in a negative way



Everything in an SOA-Based System Has To Be a Service

A service-oriented approach might not make sense for the whole system.

Guidelines for service identification are an important part of SOA strategy and governance

- Represent reusable tasks
- Have (or may have) multiple consumers
- Consumers will bind to services programmatically
- Correspond to functionality of a stateless nature
- Service inputs and outputs do not require the construction of very complex applications
- Are of a request-response nature
  - Although SOA 2.0 supports events

In Summary …

Our intent is not to discourage, but to caution

An SOA approach may be the best way to achieve common goals of interoperability, agility, and reuse

But …

- Most of the mentioned issues are active areas of research
- Some solutions will require time to mature

Also keep in mind …

- Not everything in an SOA-based system has to be a service
  - It might not make sense for your whole system
- Most case studies show that the key is governance
  - Which has very little to do with technology

Software Engineering Institute | Carnegie Mellon

Grace A. Lewis
Integration of Software-Intensive Systems (ISIS) Initiative
Senior Member of the Technical Staff
glewis@sei.cmu.edu
(412) 268-5851