Empowering Legacy Reuse with Service Oriented Architecture and Web Services

Michael Vertuno, NSD
Shan Barkataki, California State University and NSD

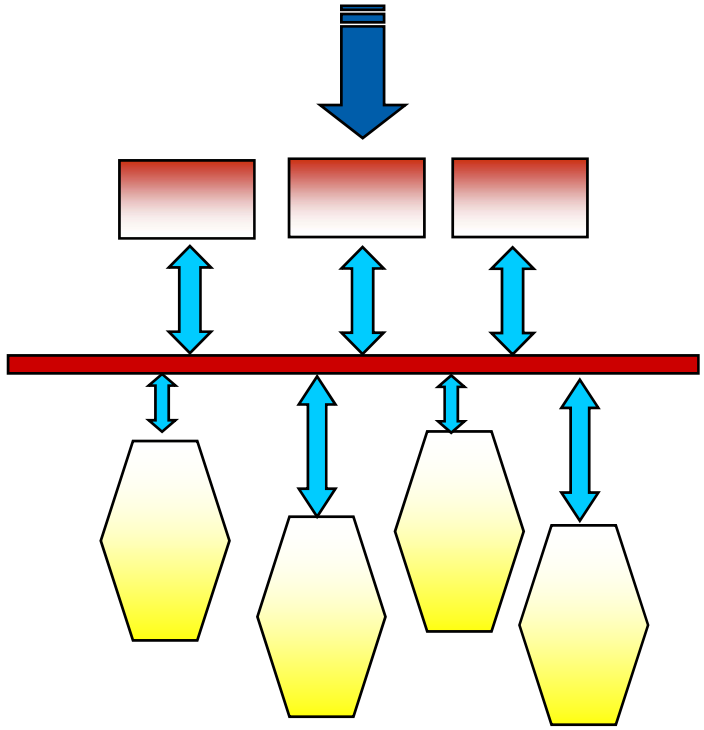
SSTC 2009
April 23, 2009

NSD: Navigation Systems Division of Northrop Grumman Corporation

NORTHROP GRUMMAN

Agenda

1. Overview
2. Classic Methods for Legacy Reuse
3. Component Based Design (Reuse and Architectural Framework)
4. Enabling Technologies
5. Binary Level Reuse
6. Overview of our Solution
7. Advantages of our Approach
8. Validation and Experience to Date
9. Summary
10. Questions and Answers



Overview

- **Classic Legacy Reuse Technique**
  - Inherited from earlier languages, platforms, and technologies
  - Represents valuable core technologies and billions of dollars of investment (time and money)
  - Parts of legacy software are routinely reused in new products and systems
  - Demands for new technologies make reuse difficult; often leading to reengineering or *copying* code
- **Motivation for Legacy Reuse**
  - Increase productivity
  - $/SLOC reduced approximately 47% by work avoidance (*IEEE source*)
  - Provide system interoperability

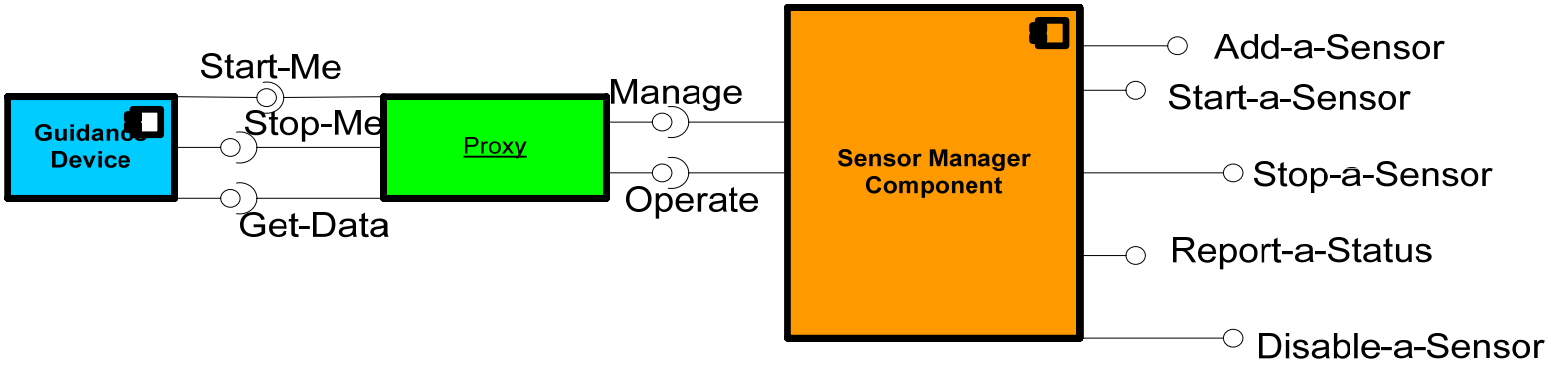
NORTHROP GRUMMAN

Classic Methods for Legacy Reuse

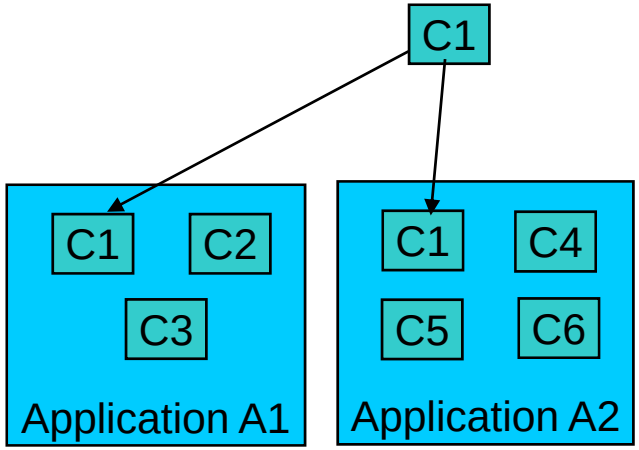
- **Predominate methods for reusing legacy code is by 'copying' code (or code duplication)**
- **Code level reuse is a poor practice**
  - Inflates volume of low-reuse legacy code
  - Must resolve interface differences during integration making it difficult
  - Resolved defects, enhancements, or other changes to the reusable code
    - Require updating multiple copies of the same source code to incorporate changes
    - Creates version control nightmare
    - Not propagated to applications
  - Does nothing to solve the reusability problem

A Better Way - Component Based Reuse

- **Make components out of legacy code**



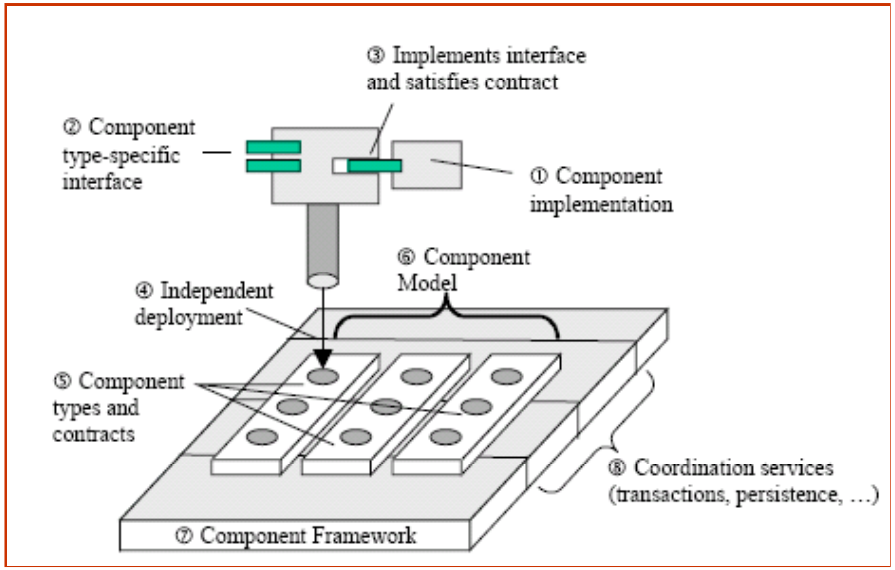
- **Build new systems with these components**
  - Reuse components across many systems
  - Achieve uniformity in design and interfaces
  - Make “whole” with “parts”



Component Based Design (CBD) and an Architectural Framework

NORTHROP GRUMMAN

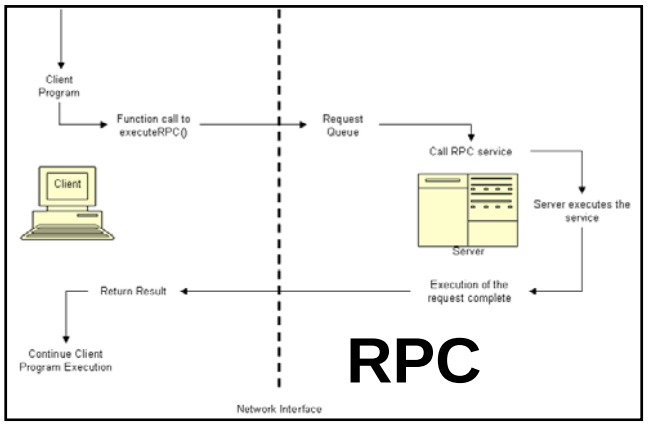
- **Critical Success Factor with CBD**
  - Using a standard software architecture
  - Conformance to component models transforms software implementations into standard architectural implementations
  - Variety of component models available - our solution uses a Service Oriented Architecture (SOA) framework

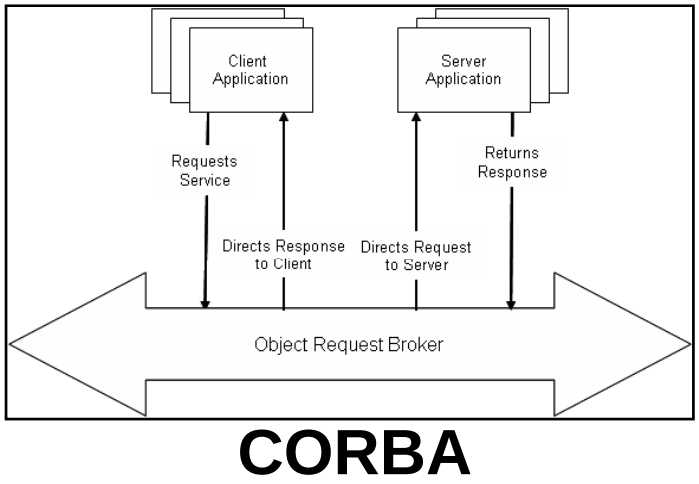


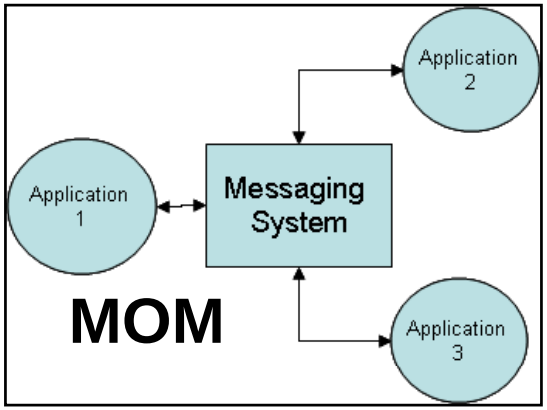
"Technical Concepts of Component-Based Software Engineering"
SW Eng Inst, Carnegie Mellon university

Enabling Technologies

- Interfacing approaches and component models specify how components interact with each other
  - Common Object Request Broker Architecture (CORBA)
  - Message-Oriented Middleware (MOM)
  - Remote Procedure Calls (RPC)
  - Microsoft DCOM and .NET
  - Web Services and J2EE



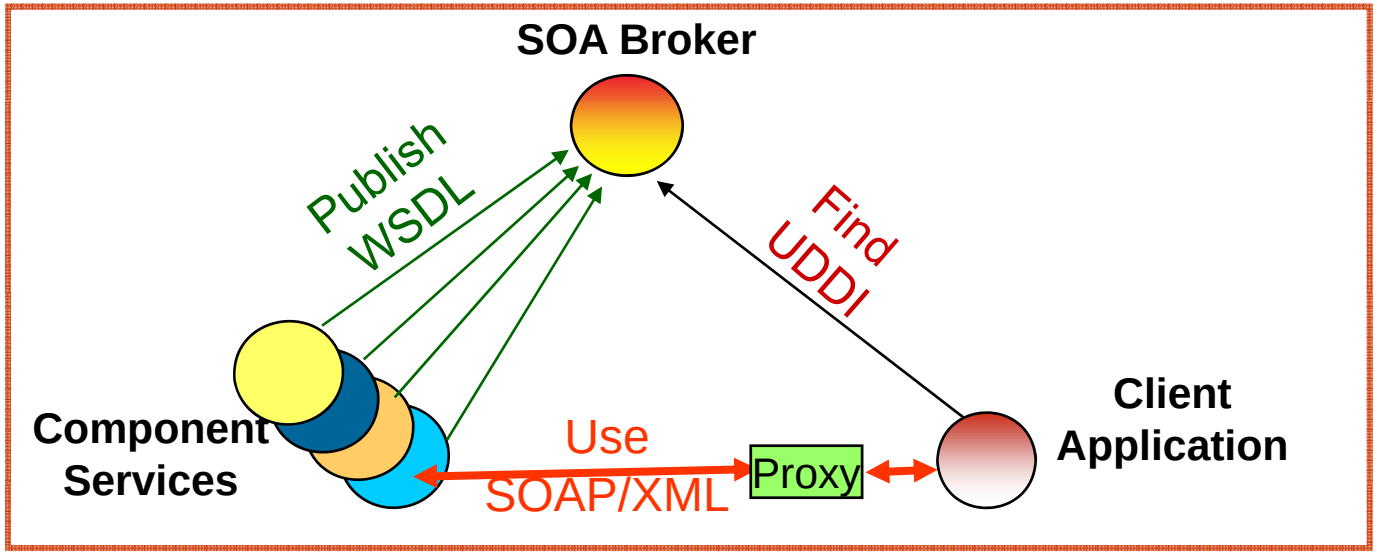




Enabling Technologies – Service Oriented Architecture (SOA)

NORTHROP GRUMMAN

- Gartner: "a style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes."
- IBM: "an application architecture within which all functions are defined as independent services with well-defined invokable interfaces which can be called in defined sequences to form business processes"
- Us: Standards based reconfiguration of legacy software with standards based interfaces. Motive = Business Agility
  - Configure legacy software into components and publish as a service in an SOA framework

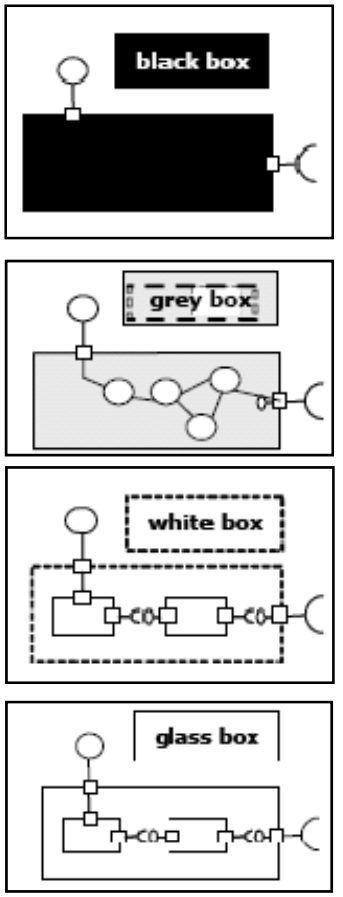


Enabling Technologies

- Service Oriented Architecture (SOA)
  - Principles include encapsulation, abstraction, reusability, autonomy, optimization, discoverability
  - Service provider exposes its interface and information necessary to use the service
  - Web Services implement a SOA
    - SOAP – protocol to exchange XML messages commonly using HTTP; but not necessary
    - More flexible over other interfacing approaches
- Resolve by linking vs. Resolve dynamically?
  - UDDI - Universal Description, Discovery and Integration
  - Discover servers dynamically

Binary Level Reuse

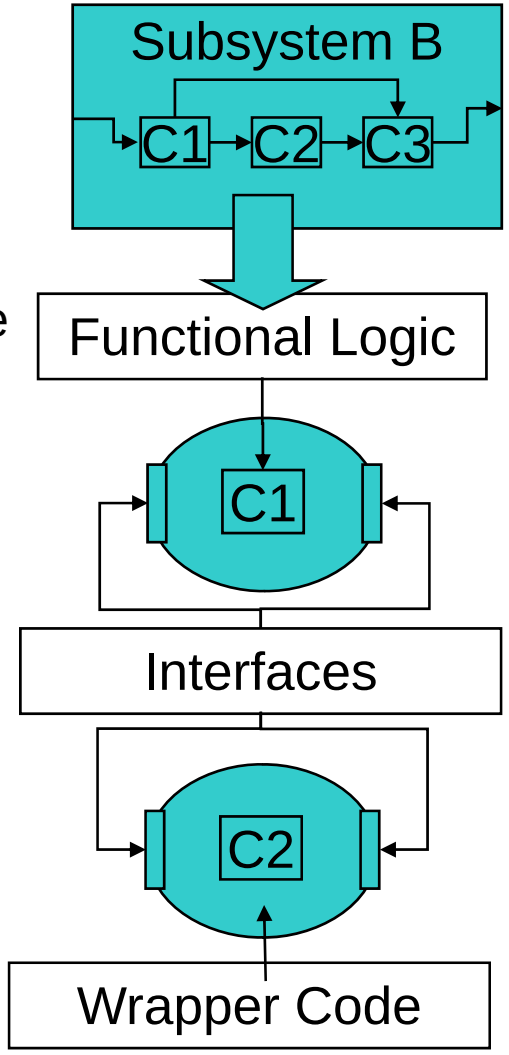
- **Binary (or library) level reuse**
  - Facilitates reuse of the same compiled code from multiple projects without needing access to the source code (examples: Windows DLLs & COM objects)
- **Advantages**
  - Source code and binary version control issues are centralized and not distributed across all users maintaining specialized versions of source or binary
  - Although source code is not available, users can still extend the functionalities of the reused components using wrappers and facades
  - Facilitates Black box reuse of legacy code
    - Turn legacy code into binary reusable components
    - Group related functions into a single component
    - Modularization - make components reusable across different systems
  - Components are “black boxes” to its clients
    - Implementation detail is hidden
    - Interaction via messages



“Component-based Approach and Dependable Systems”
Dept of CS/Eng, Malardalen University

Overview of Solution

- **Basic Methodology**
  - Extract reusable components (C1 and C2) from Legacy Subsystem B
  - Wrap reusable components
  - Develop public interface, protocol, and required message formats to interact with components
    - Platform independent data types
- **C1 and C2 are reusable components**
  - Can be independent or create another composite component
  - Subsystem B as well as other systems can request their services



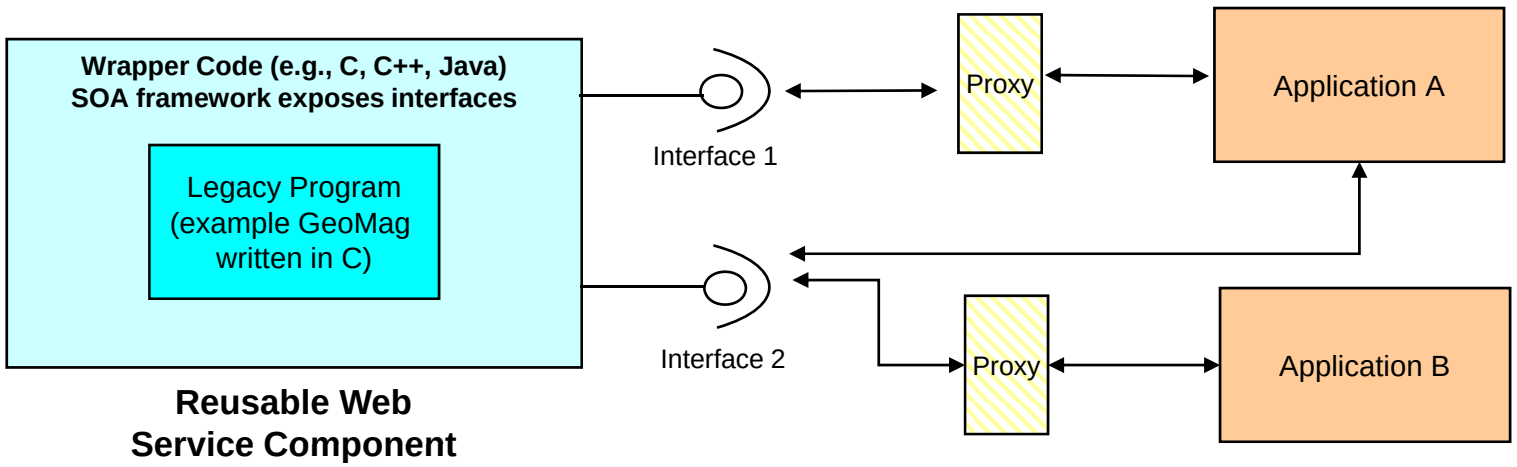
Overview of Solution - Component Reuse and Extracting Interfaces

NORTHROP GRUMMAN

- **Extraction**
  - Identify code of a business operation to determine feasibility of reuse as a public interface
    - Use control and data flow analysis to identify processing and affected data objects.
  - SoftLink process (*Sneed, Harry; Fifth IEEE International Workshop, 2003*)
    - Process for creating Web Services from Legacy Applications
    - Function mining, function wrapping, XML schemas
  - Brute force extraction of reusable functional components from the legacy application
- **Recompose interfaces**
  - Wrap function prototypes in a generic fashion
  - Interface definitions describe how components should be utilized
    - Must be clear, unambiguous, platform independent
    - Web Service Description Language (WSDL)
  - Resolve interface differences with a ‘proxy’ component

Example Reuse Architecture

- Example using a legacy application (GeoMag) written in C that performs various Geodetic calculations, including magnetic variation
- Wrapper code exposes two interfaces (web services)
  - Application A uses both interfaces 1 and 2
  - Application B uses interface 2
- Proxies are used to resolve interface differences



Sample Interface Specification for GeoMag Example

NORTHROP GRUMMAN

- Interface for calculating magnetic variation
- Defines inputs and outputs
  - Input: Lat/Long, Altitude, Date
  - Output: Magnetic Variation
- Written in WSDL

GeoMagWSDLOperationRequest (5 parts)

Part Name	Part Element or Type
latitudeDecimalDegrees	xsd:double
longitudeDecimalDegrees	xsd:double
altitudeInHundredsOfMeters	xsd:float
yearInteger	xsd:positiveInteger
monthInteger	xsd:positiveInteger

Add Part | Remove Part

GeoMagWSDLOperationReply (1 part)

Part Name	Part Element or Type
magneticDeclination	xsd:double

Add Part | Remove Part

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="GeoMagWSDL" targetNamespace="http://j2ee.netbeans.org/wsdl/GeoM
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://j2ee.netbeans.o
    <types/>
    <message name="GeoMagWSDLOperationRequest">
        <part name="latitudeDecimalDegrees" type="xsd:double">
            <documentation>Latitude In Decimal Degrees (-90 to 90)</documentation>
        </part>
        <part name="longitudeDecimalDegrees" type="xsd:double">
            <documentation>Longitude in Decimal Degrees (-180 to 180)</documentati
        </part>
        <part name="altitudeInHundredsOfMeters" type="xsd:float"/>
        <part name="yearInteger" type="xsd:positiveInteger"/>
        <part name="monthInteger" type="xsd:positiveInteger"/>
    </message>
    <message name="GeoMagWSDLOperationReply">
        <part name="magneticDeclination" type="xsd:double"/>
    </message>
    <portType name="GeoMagWSDLPortType">
        <operation name="GeoMagWSDLOperation">
            <input name="input1" message="tns:GeoMagWSDLOperationRequest"/>
            <output name="output1" message="tns:GeoMagWSDLOperationReply"/>
        </operation>
    </portType>
    <binding name="GeoMagWSDLBinding" type="tns:GeoMagWSDLPortType">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"
        <operation name="GeoMagWSDLOperation">
            <soap:operation/>
            <input name="input1">
                <soap:body use="literal" namespace="http://j2ee.netbeans.org/wsdl/
            </input>
            <output name="output1">
                <soap:body use="literal" namespace="http://j2ee.netbeans.org/wsdl/
            </output>
        </operation>
    </binding>
    <service name="GeoMagWSDLService">
        <port name="GeoMagWSDLPort" binding="tns:GeoMagWSDLBinding">
            <soap:address location="http://localhost:18181/GeoMagWSDLService/GeoMa
        </port>
    </service>
    <plnk:partnerLinkType name="GeoMagWSDL1">
        <!-- A partner link type is automatically generated when a new port type i
In a BPEL process, a partner link represents the interaction between the BPEL proc
A partner link type characterizes the conversational relationship between two serv
        <plnk:role name="GeoMagWSDLPortTypeRole" portType="tns:GeoMagWSDLPortType"
    </plnk:partnerLinkType>
</definitions>
```

NORTHROP GRUMMAN

Advantages of our approach

- **Programmers do not need to know implementation details**
  - Only knowledge of the interface is required
- **Transforms software implementations into architectural implementations (service oriented)**
- **Interoperable between systems and programming languages**
- **Components can be interconnected**
  - A component may rely on many other components
- **Independent deployment**
  - Dependences can be resolved at runtime
  - ‘Broken’ components can be replaced or substituted minimizing system downtime

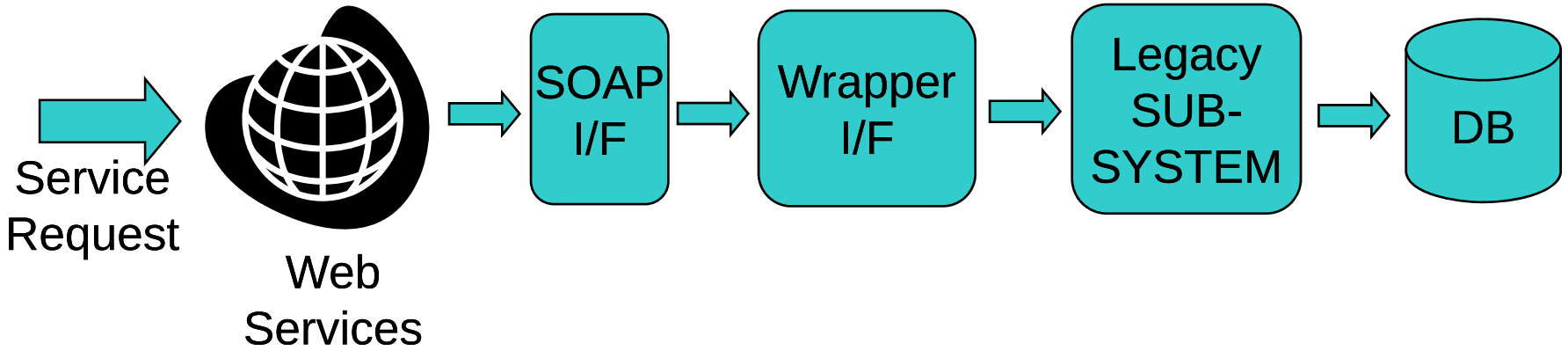
Validation Through IRAD Prototypes

- The concept has been demonstrated in an IRAD project at NSD and used in several applications involving many different functionalities
  - Interfaced open architecture Air Defense System with Air Battle Management System
    - Integrated non-real time data into real time system
    - Mission Planning, Air Combat Order (ACO), Air Tasking Order (ATO)
  - Integrated ground picture into air picture
    - Red/blue force tracking
    - Enhanced situational awareness

Experience to Date
In IRAD Projects

NORTHROP GRUMMAN

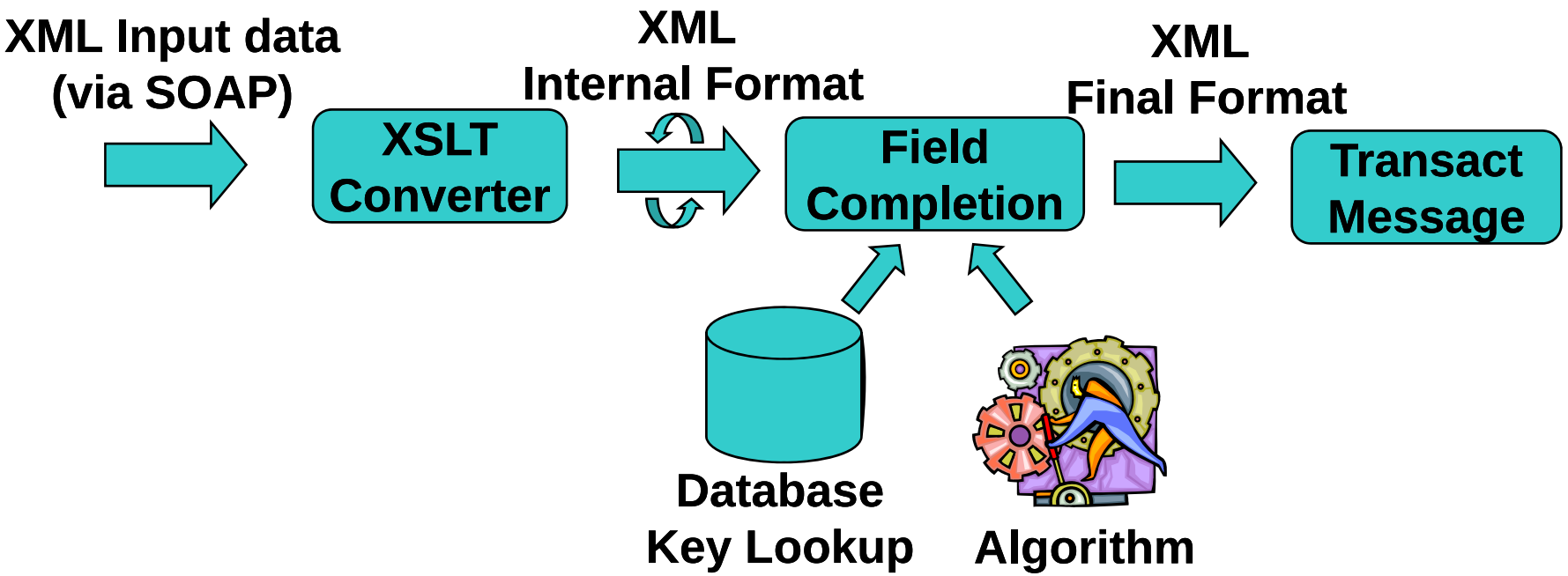
- Developed proxy component for Web Service Interface to open architecture air defense system
- Design Goals
  - Create a robust interface
  - Minimize infrastructure changes to connect system
  - Base the data exchange on XML
  - Support XML API - convert to legacy internal format
- Design Approach
  - Build the interface on SOAP Web services
  - Pass XML data via SOAP transactions



Experience to Date
In IRAD Projects

NORTHROP GRUMMAN

- SOAP Interface Component
  - Interface acts as a proxy
  - Extensible Stylesheet Transformations (XSLT) convert XML to internal format
  - Database and algorithms assist in resolving interface differences (e.g., units, defaults, enumerations)



Experience to Date In IRAD Projects

NORTHROP GRUMMAN

- Web Services within Command and Control (C2)
  - Clients can communicate via Web Service to access Track, Unit Task Organization (UTO), and Overlay data
  - Clients are able to create, modify, and delete Track and UTO data
  - Schemas
    - Exchange Track information using Global Command and Control System (GCCS) Tactical Management System (TMS) Track schema
    - Exchange Overlay information using the Defense Information Systems Agency (DISA) XML Overlay schema
- Successfully used Web Service to integrate ground picture from Web Service with air picture
  - Exchange tactical picture in near real time
  - Red/blue force tracking
  - Link-16 messages

Future Considerations

- True web service architecture in a net centric environment
- Dynamic binding

NORTHROP GRUMMAN

Summary

- Software reuse has many benefits, specifically at a binary level
- Emerging technologies (e.g., CBD, SOA) can assist in a developing a structured and disciplined architecture for component reuse
- Distributed systems with network centric architectures would greatly benefit by reusing the wealth of legacy code.

Contact information:

Michael Vertuno	mike.vertuno@ngc.com	818.712.7402
Shan Barkataki	shan.barkataki@csun.edu	818.677.3398

Acronyms

- ACO – Air Combat Order
- API – Application Programming Interface
- ATO – Air Tasking Order
- C2 – Command and Control
- CBD - Component Based Design
- COM – Component Object Model
- CORBA – Common Object Request Broker Architecture
- DISA - Defense Information Systems Agency
- DLL – Dynamic Link Library
- GCCS - Global Command and Control System
- HTTP – Hypertext Transfer Protocol
- IEEE - Institute of Electrical and Electronics Engineers
- IRAD – Internal Research and Development
- J2EE - Java 2 Platform, Enterprise Edition
- MOM – Message-Oriented Middleware
- NSD – Navigation Systems Division of Northrop Grumman Corporation
- RPC – Remote Procedure Call
- SOA – Service Oriented Architecture
- SOAP – Simple Object Access Protocol
- SLOC – Source Lines of Code
- TMS – Tactical Management System
- UDDI – Universal Description, Discovery and Integration
- UTO - Unit Task Organization
- XSLT – Extensible Stylesheet Language Transformation
- XML – Extensible Markup Language
- WSDL – Web Service Definition Language


NORTHROP GRUMMAN
DEFINING THE FUTURE
Questions