

Model Driven Code Generation

Andy Farrar

SAIC

farrarm@saic.com

William Ray

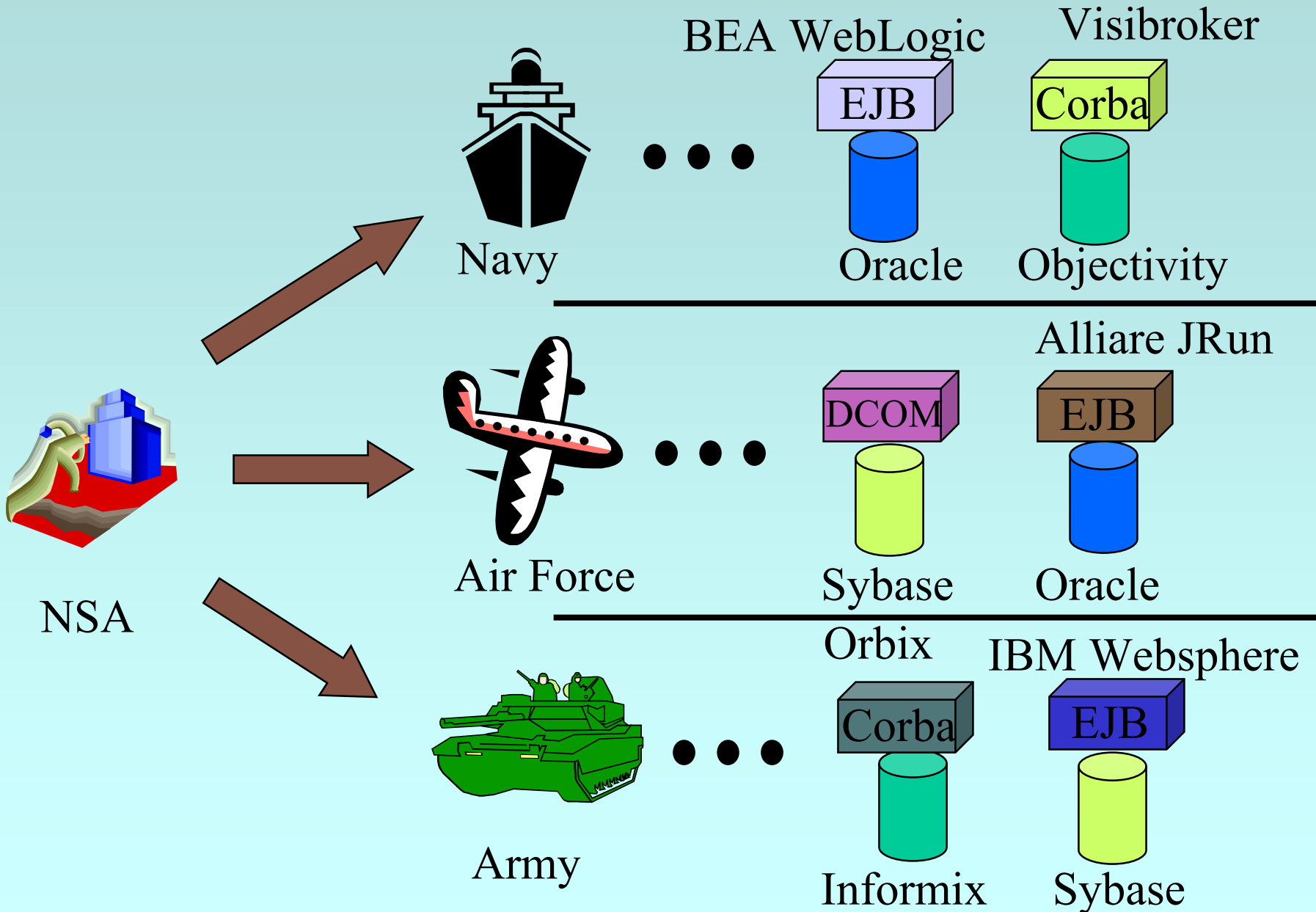
SPAWAR

ray@spawar.navy.mil

Outline

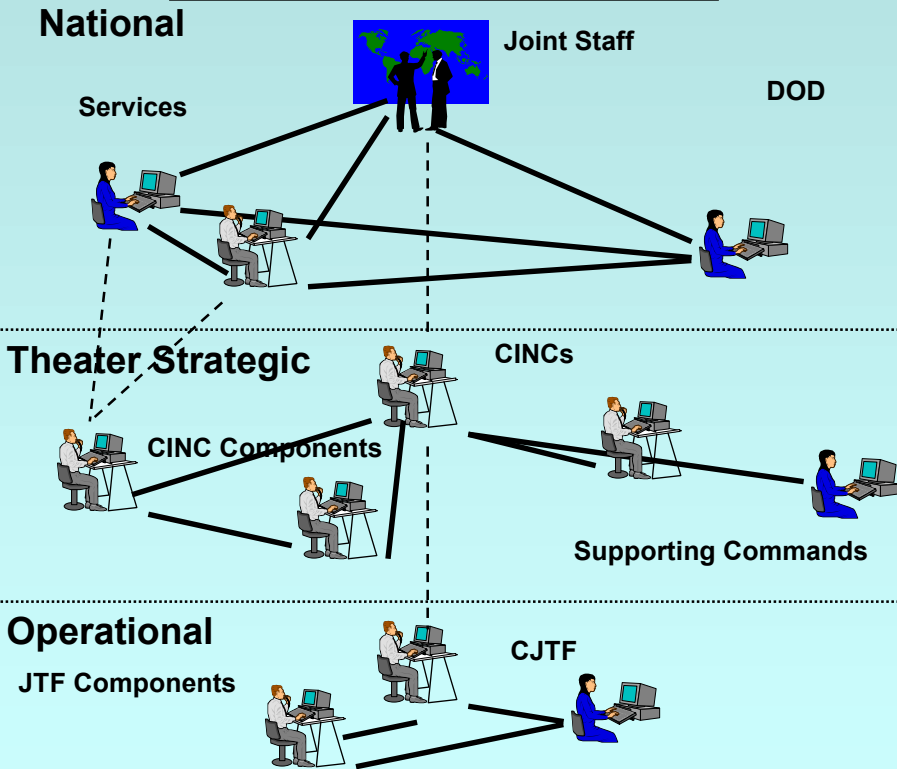
- *Motivation*
- *Approach*
- *Results*
- *Code Generation Architecture*
- *Examples*
- *Animation & Demo*
- *Conclusion*

Motivation

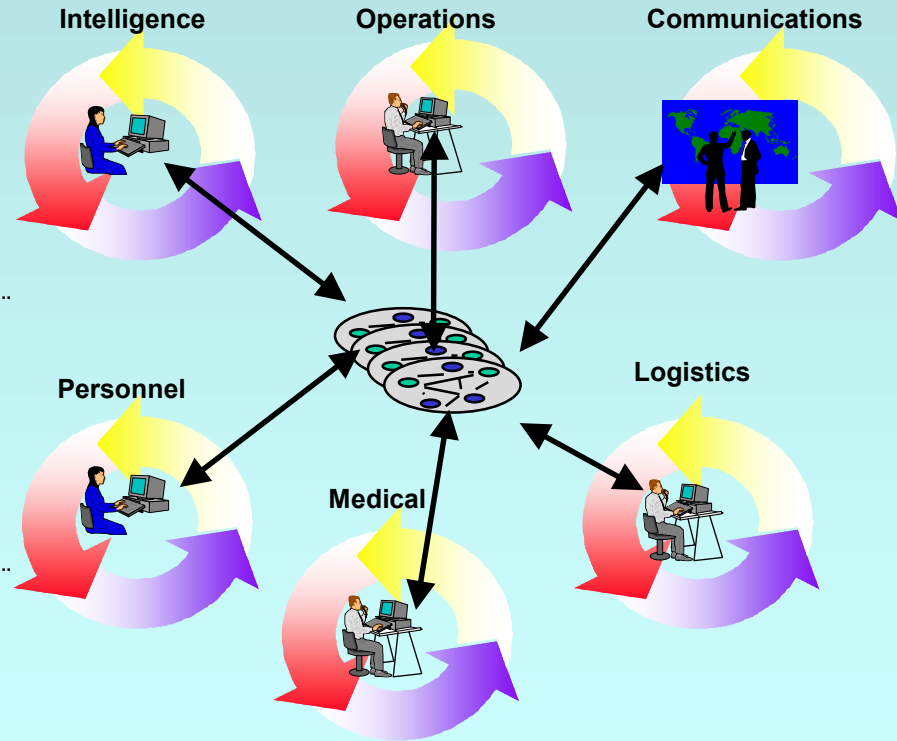


Operational Architecture

Sharing Data Across all Echelons



Sharing Data Across Multiple Domains



Information Assurance
(SECURITY, AVAILABILITY, ACCURACY)

Motivation

- **Distributed Systems are Hard to Build**
 - Too much time is spent building the middle-tier
 - Leverage enterprise knowledge (Force Multiplier)
 - Experts are needed for design and implementation
 - Thousands of application developers
 - Heterogeneity across the enterprise
- **Technology Evolution makes Selection of Middleware Difficult and Costly**
 - Market moves to “Newer” technology, leaves developers and clients with stagnate and under supported technology
 - Difficult to make cost effective use of the latest advances in technology

Motivation

- ***Composability of Services and Components***
 - Enterprise Java Beans (EJBs), CORBA Services, COM Components, and Web Services
 - Too much work required of developers to make use of services
 - Need quick way to plug in enterprise components

Approach

- **Maximize use of metadata**
 - Focus on the use of UML with/without extensions
 - Provide the bridge to go directly from model to code
 - Provide importers for the use of various modeling tools
- **Balance process with rapid development**
 - Provide traceability from model to code and back
 - Provide for rapid development with changing object models
- **Provide an open environment that could be used for generation of needed logic and supporting files**
 - Support both scripting and OO programming styles for code template definitions

Results

- **Provided a repeatable process for software development**
 - Aligned with the Rational Unified Process
 - Consistent with CMM

- **Reduced or eliminated manual programming for supporting methods**
 - Reduced need for maintenance
 - Allowed rapid switching between different middleware technologies
 - Provided support for rapid schema evolution

Results

- **Made more effective use of scarce software development talent**
 - Specialized engineers created templates to support different system frameworks and services, allowing less experienced developers to rapidly produce working implementations
- **Provided an integration point for new technologies**
 - Allowed developers to quickly evaluate new components, services, or alternate middleware technologies
 - Easily Switched between different middleware technologies
 - CORBA, COM+, EJB, Agent, Web Services, etc.

Results

- **Model to Model mappings provided additional metadata capture and use**
 - Logical to physical mappings provided automated database support
 - Database definition, JDBC persistence generation, database join information
 - Logical to logical mappings provided for automated translation using various technologies
 - XSLT, interface engines, etc...

Composability

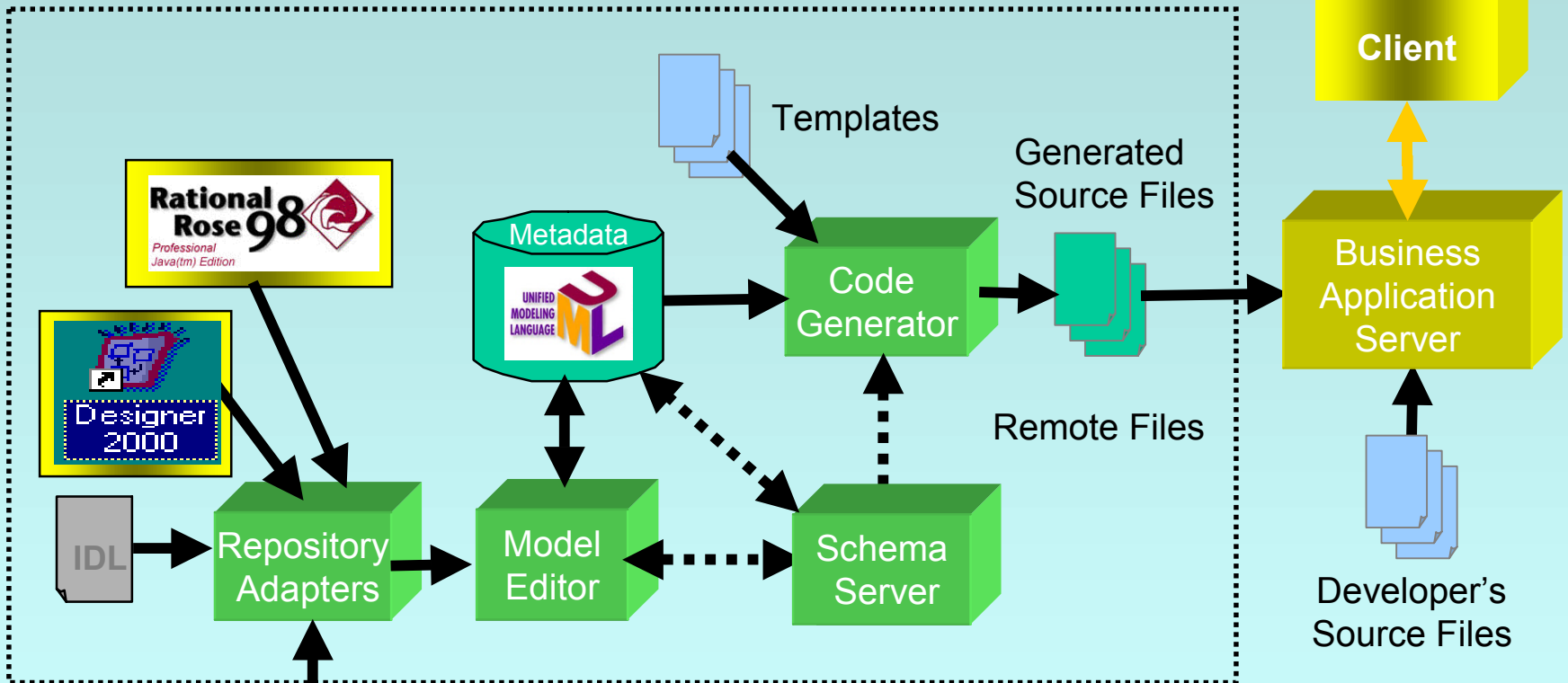
- **Define the composability aspects of the services or components**
 - Components require glue code
 - Services require supporting code at the class level
- **Decorator pattern was used to minimize the affects of template on the others**
- **Templates can use any type of language**
 - Scripting language
 - Difficult to determine composibility
 - Java
 - Much easier to define composibility using interface definitions

System Components

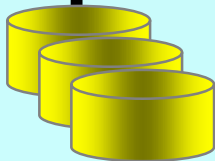
- *Integrated Generation Environment*
- *SchemaServer*
 - Enterprise Accessible Model Repository
 - Repository Adapters for import from different sources
- *Code Generator*
 - ECMAScript templates
 - Java templates

Components - Scenario


Provides the Information Infrastructure





Information
Repositories
Oracle,
Sybase, etc



LEGEND

System Component = 

COTS Component = 

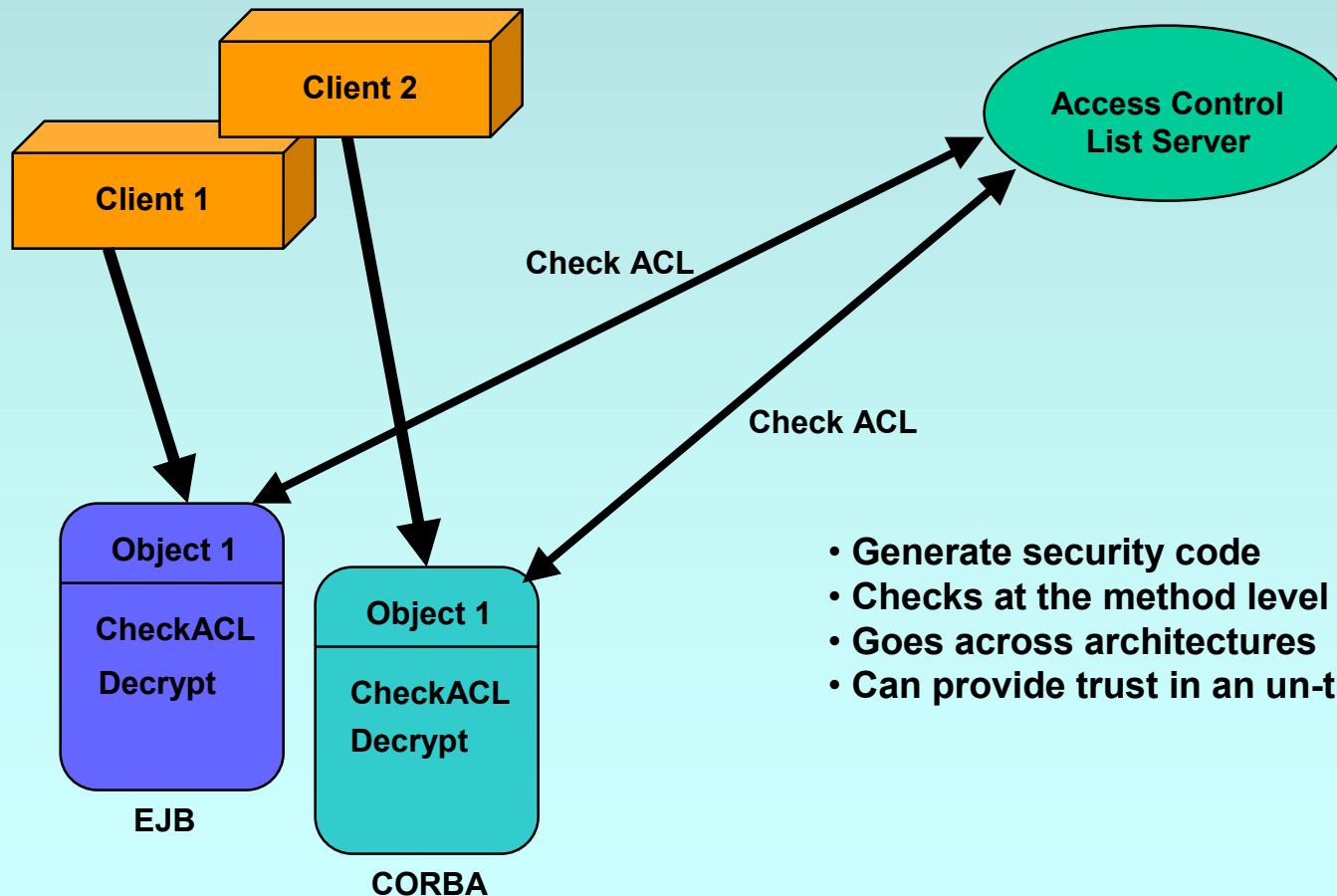
Hand-Coded Component = 

Simple Generation Examples

- **Source code – any language**
- **Properties files**
- **Test clients**
- **Front end support**
 - *HTML, JSPs, or ASPs*
- **XML Support**
 - *DTDs, XML-Schema, XSLT, XML Streaming, etc..*

Security Example

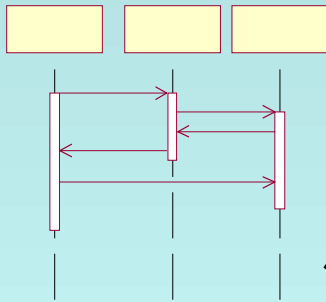
- **Object level security**



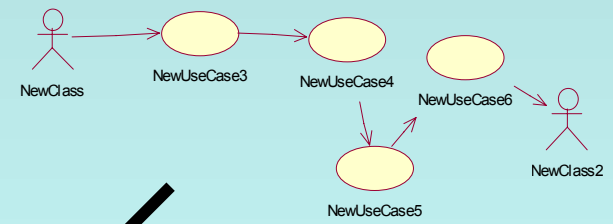
- Generate security code
- Checks at the method level
- Goes across architectures
- Can provide trust in an un-trusted environment

Expanded use of UML Example

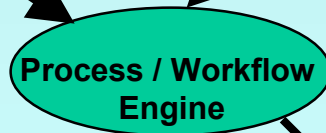
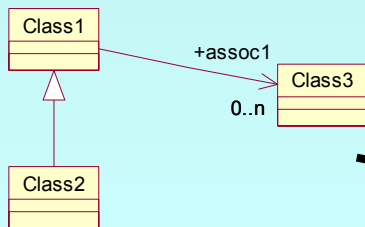
Sequence Diagram



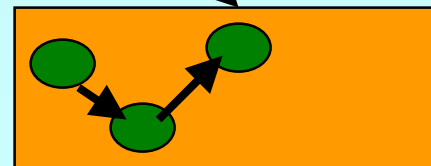
Use Case Diagram



Class Diagram



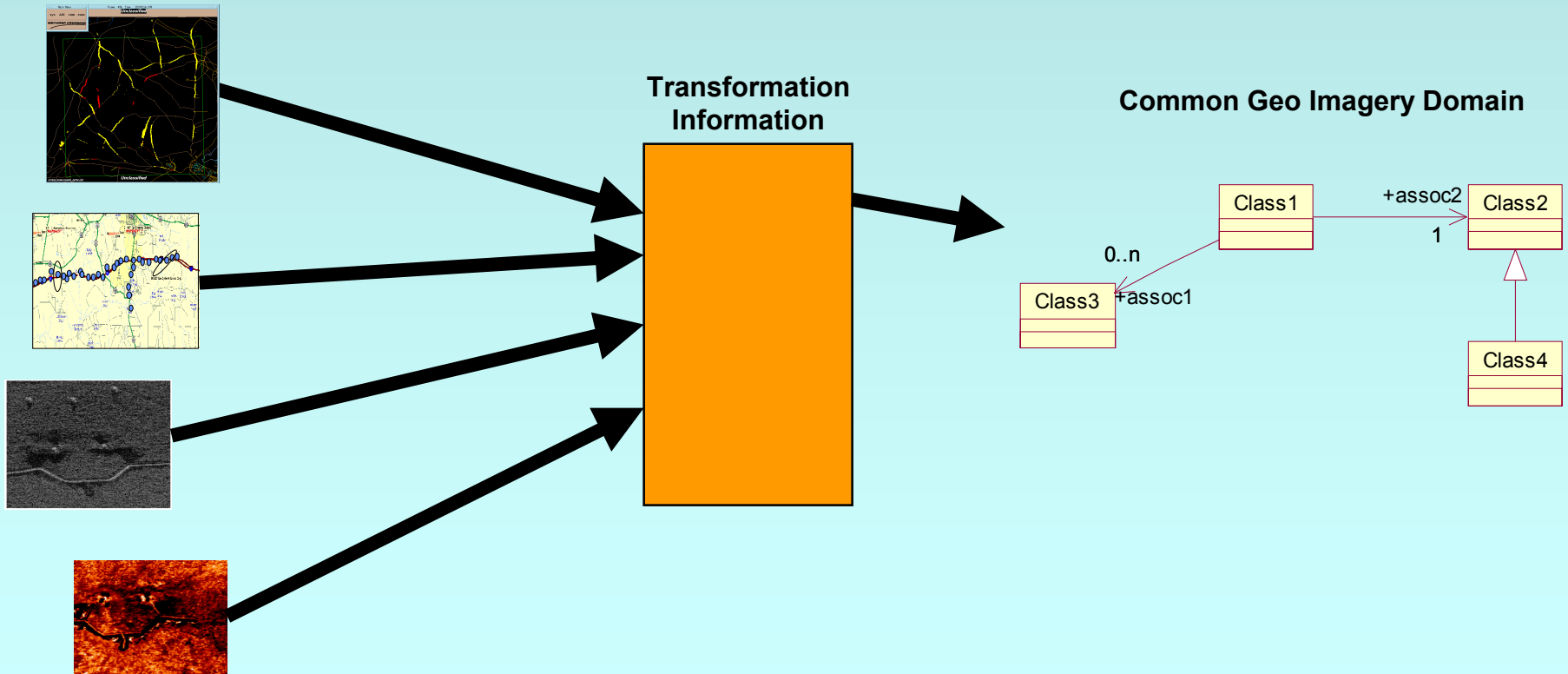
- 1) Clinical Trial Process flow
- 2) Combined use of multiple diagrams to provide both generated classes and input to process/workflow engine



Executing System

Model 2 Model Mapping Example

Model to Model Mappings using Refinements with conditions to provide the metadata to generate the necessary mapping code in various forms (Java, XSLT, etc...)



J2EE Generation Examples

- **Additional logic required by system design**
 - Security services
 - System services
 - Bean managed persistence
 - Interfaces to other architectures
 - Agents architectures, CORBA, COM
 - Management methods
- **Java Classes required by J2EE**
 - Home, Remote, Impl classes
- **Deployment Descriptors**
 - The standard descriptors
 - Ejb-jar.xml, web.xml, application.xml
 - Vendor specific descriptors
 - Weblogic.xml, ias-web.xml, etc...

CORBA Generation Examples

- **Additional logic required by system design**
 - Security services
 - CORBA Service support
 - LifeCycle, Persistence, Relationship, etc..
 - Interfaces to other architectures
 - Agents architectures, COM
 - Management methods
- **Different CORBA design patterns**
 - Direct inheritance from CORBA object
 - Delegation pattern
- **Support for both BOA and POA ORB approaches**

Case Study Examples

Case	Number of Classes	Average Attributes per Class	Average Operations per Class	Lines of Code Generated
A	7	30	24	8,000
B	118	22	6	257,000
C	321	12	8	750,000

Note: Lines of code vary with the number of services enabled for that run of code generation

Specifics on Metrics

Small Example: DoD “planning” schema with 7 classes, 30 attributes, and 24 methods (JFACC ISTI-98)

Hand Coded

- 3 weeks for dev.
- 3640 lines of code.
- Simple CORBA Server.

Supplemented with Code Generator

- 12 hours for development.
- 1163 lines of user defined code.
- 8037 generated lines of code.
- CORBA Server with services supporting Naming, Lifecycle, Externalization, and Proxy Objects.

Specifics on Metrics

- **Advanced Courses Of Action Project - ACTD**
 - **8 Projects months saved.... (PM Estimate)**
 - Significant changes to model - 12 times in 9 months
 - **Over 750,000 lines of source code generated supporting both the backend servers and the front end user interfaces.**
 - ~250,000 lines of front end GIS and application support code
 - ~500,000 lines of backend CORBA code supporting lifecycle, collaboration, RDBMS persistence/OODBMS persistence, alerts/messaging, and logging

Conclusion

- **Work successfully produced a set of tools for rapid system prototyping and development**
- **Tools were found to be useful by multiple projects (more everyday...)**
- **Tools provided a mechanism to experiment supporting rapid development**
 - **Forms of composability of services and components**
 - **Middleware technologies**

Needed Research Areas

- **Prove template correctness**
- **Prove template combination correctness**
- **Combining ADL extensions to UML for additional generation logic**