Understanding Object Oriented Software

Melissa P. Chase

781-271-7320 • drh@mitre.org

MITRE Sponsored Research

MITRE Technology Program

Problem

- Developing and sustaining large, reliable software systems is a challenge
- Understanding software assets is necessary for decreasing the cost and improving the quality of software by supporting
  - Iterative development, training
  - Reuse of legacy components
  - Assessment of software quality, performance, security, and other features

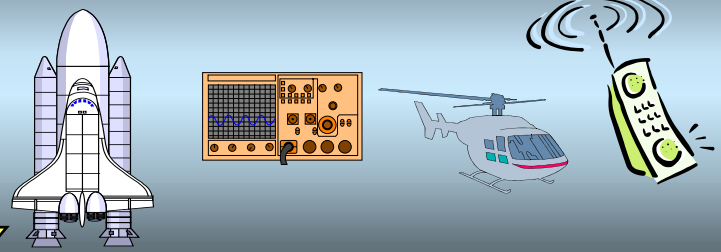
MITRE

Background

Software Developers

Using
S/W architecture
Design Patterns
Programming Idioms

Large, Complex Systems



Source Code (today's systems using object-oriented software - C++, Java)

What have we got here?

How and where can we safely and effectively use it?

Analysts, Funding agencies

?????

If we know the structures used, can we answer these questions?

MITRE

Objective

- Develop the capability to understand object oriented software by
  - Recognizing patterns of communicating classes (design patterns and architecture patterns), and
  - Reasoning about these interacting classes to understand the rationale behind design decisions

MITRE

Activities

- Develop class interaction extractors for C, C++ programs
- Build small-footprint analysis tools -- Osprey
  - Specification language for design pattern recognizers. 68 recognizers for 41 patterns
  - Finders that match source code data to recognizer descriptions
  - Generator for pattern-centric documents -- inferred design, code qualities, links to source code
- Validate results using Go4, SUIF, ACE, Mozilla -- documented use of patterns

MITRE

Highlight

TCL Wrapper

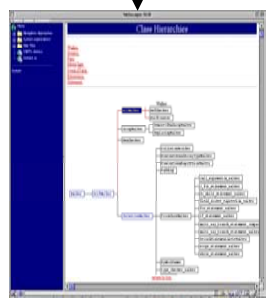
Imagix

High-level Analysis
- Compliance
- Pattern interactions
- Trade-offs
- Design Rationale

Design and Architecture Pattern Recovery

Recognizer Library

Visualization and Presentation



Class hierarchy
Variable set/use
Call trees

```
<class> Observer
<method> update </method>
<member> observerState
<type> Subject-ref </type
</member>
</class>
```

XML Database

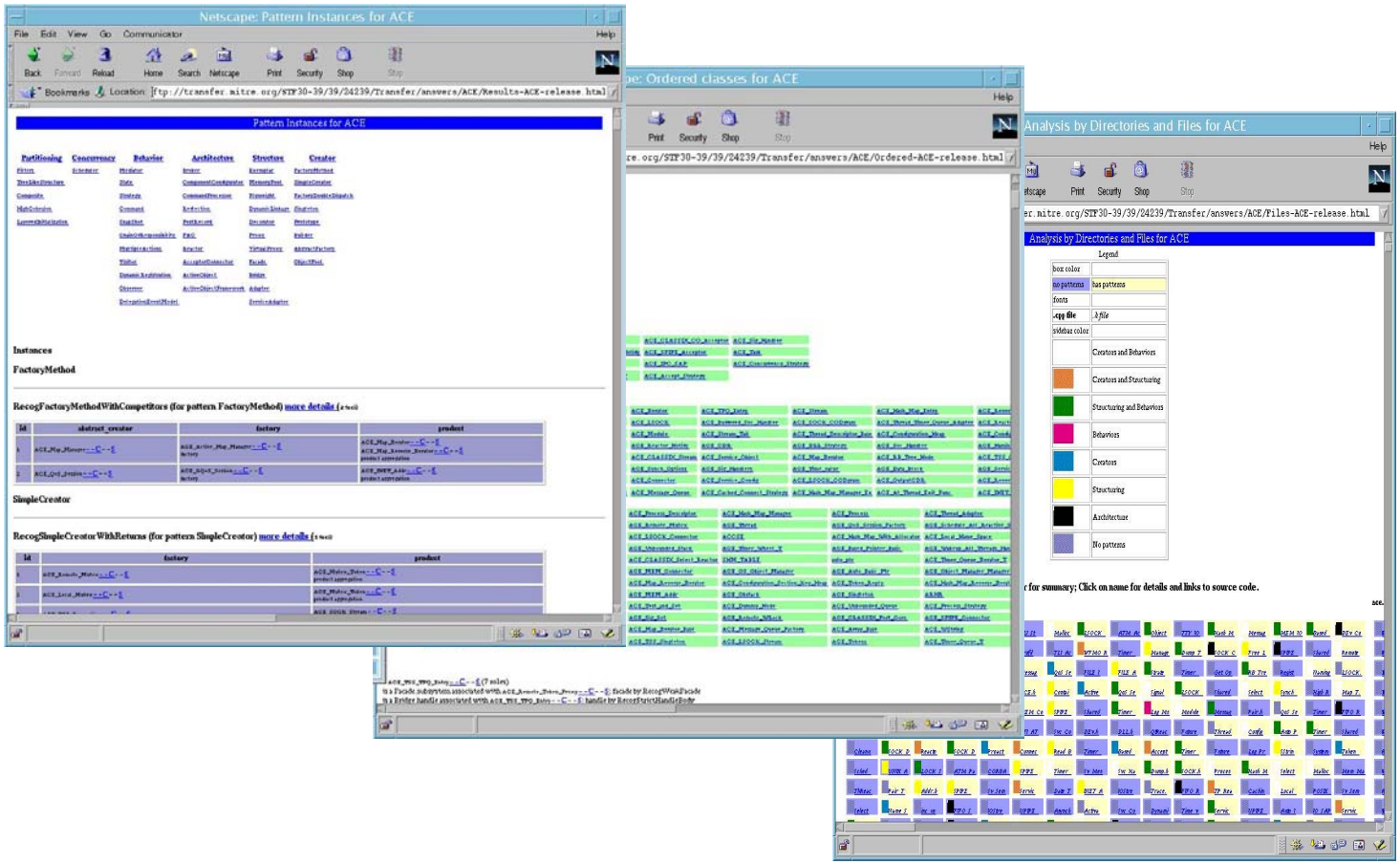
Low-level Analysis
- Class interactions
- Type inference
- Control Flow
- Data Flow

Osprey Architecture

MITRE

Demonstration

Osprey-Generated Documentation Packages

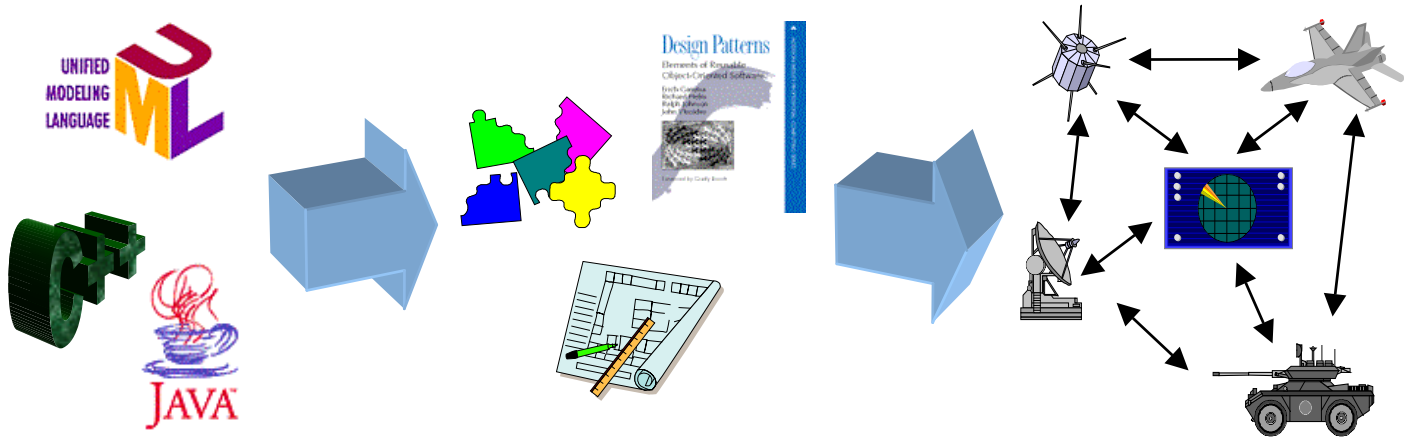


Impacts

- Support for Integrated Broadcast Service (IBS) program - quality assessments
- Analysis of MITRE-developed MSIM simulation program - documentation
- Developing software vulnerability recognition for C++ code
  - Internal and external requests for this capability

MITRE

Future Plans



Using
Source code
Models
Documentation

Recognize
Patterns
Components
Architectures

Map design elements to functionality
Analyze design trade-offs
Ensure architectural conformance
Assess qualities (e.g., performance, security)

MITRE